(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2001 (22.03.2001)

PCT

(10) International Publication Number
WO 01/20562 A2

(51) International Patent Classification[7]: G07C 13/00

(21) International Application Number: PCT/US00/07737

(22) International Filing Date: 24 March 2000 (24.03.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/126,080 25 March 1999 (25.03.1999) US
60/149,621 16 August 1999 (16.08.1999) US

(71) Applicant *(for all designated States except US)*: **VOTE-HERE. NET** [—/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **ADLER, Jim** [—/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US). **GREEN, Richard, L.** [—/US]; 190 Lyme Road, Hanover, NH 03755 (US). **NEFF, C., Andrew** [—/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US). **DAI, Wei** [—/US]; 13440 SE 24th Street, Bellevue, WA 98005 (US).

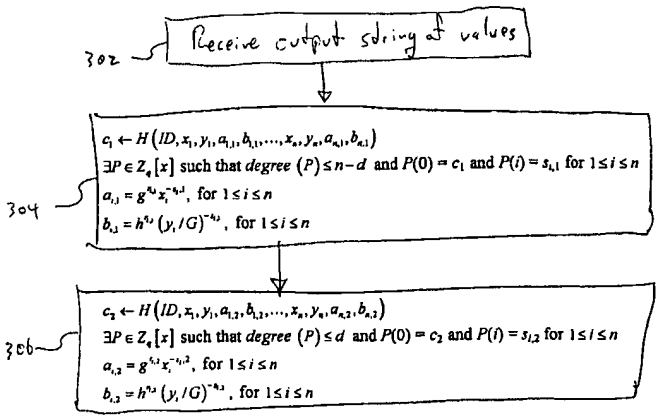
(74) Agents: **DALEY-WATSON, Christopher, J.** et al.; Perkins Coie LLP, Suite 4800, 1201 Third Avenue, Seattle, WA 98101-3099 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: MULTIWAY ELECTION METHOD AND APPARATUS



(57) Abstract: A scheme or protocol for computationally secure, practical electronic voting is disclosed that supports multiple ballot options and write-in options. The scheme is based on homomorphic threshold cryptographic schemes. The scheme is efficient in computing election tallies for multi-way races (*i.e.*, for choosing *d*-of-*N* options). In addition to implementation in the discrete log context, the scheme is presented in an elliptic curve setting.

WO 01/20562 A2

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

MULTIWAY ELECTION METHOD AND APPARATUS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application Nos. 60/126,080 and 60/149,621, filed March 25, 1999 and August 16, 1999, respectively, both currently pending.

TECHNICAL FIELD

The following relates generally to encryption, and more specifically to electronic encryption employed in voting schemes.

BACKGROUND OF THE INVENTION

Requirements for an electronic voting scheme represent one of the most challenging in secure multi-party computation. Privacy of the voter must be protected, strict audit trails must be in-place, and the system must be easy to use and administer. Furthermore, ballot types range from simple *yes/no* initiatives to complex multi-way candidate races allowing the possibility for write-in candidates.

Considerable work has been done to develop a secure and efficient electronic voting scheme. These approaches fall into two categories: (1) those based on anonymous channels and (2) those based on number-theoretic techniques without anonymous channels.

Schemes that use anonymous channels to protect the voter's privacy are described in D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, 24(2):84-88, 1981; A. Fujioka, T. Okamoto, K. Ohta, "A practical secret voting scheme for large scale elections," Advances in Cryptology - AUSCRYPT '92, Lecture Notes in Computer Science, pp. 244-251, Springer-Verlag, 1992; and C. Park, K. Itoh, K. Kurosawa, "Efficient anonymous channel and all/nothing election scheme," Advances in Cryptology - EUROCRYPT '93, Lecture Notes in Computer Science, pp. 248-259, Springer-Verlag, 1993. These schemes are

efficient and allow arbitrary ballot types (*yes/no*, *d*-of-*N* options, write-ins). However, the most significant disadvantage of such schemes is that the voter is responsible for insuring that their vote was accounted for in the final election tally. This "individual verifiability" property is highly impractical for public-sector elections. Essentially, no independent observer can verify the election.

Number-theoretic schemes that do not use anonymous channels (homomorphic schemes) are described in J. Benaloh, "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret," Advances in Cryptology - CRYPTO '86, Lecture Notes in Computer Science, pp. 251-26O, Springer-Verlag, Berlin 1987; J. Benaloh, M. Yung, "Distributing the power of a government to enhance the privacy of voters," ACM Symposium on Principles of Distributed Computing, pp. 52-62, 1986; K. Sako, J. Kilian, "Secure voting using partially compatible homomorphisms, Advances in Cryptology-CRYPTO '94, Lecture Notes in Computer Science, Springer-Verlag, 1994; R. Cramer, M. Franklin, B. Schoenmakers, M. Yung, "Multiauthority secret-ballot elections with linear work," Advances in Cryptology - EUROCRYPT '96 Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1996; R. Cramer, R. Gennaro, B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," Advances in Cryptology - EUROCRYPT '97, Lecture Notes in Computer Science, Springer-Verlag, 1997; U.S. Patent No. 5,495,532, J. Kilian, K. Sako, "Secure electronic voting using partially compatible homomorphisms," issued 2/27/1996, filed 8/19/1994; U.S. Patent No. 5,682,430, J. Kilian, K. Sako, "Secure anonymous message transfer and voting scheme," issued 10/28/1997, filed 1/23/1995. These schemes enjoy a "universal verifiability" property where anyone can verify the election without compromising voters' privacy. The system under Cramer et al., employs threshold cryptosystems to achieve robustness in the face of faulty authorities. The systems under Benaloh and Sako/Kilian employ (verifiable) secret-sharing to achieve the same goal but at greater computational cost to the voter. However, these schemes do not necessarily support arbitrary ballot types and suffer from higher computational requirements. The two categories for electronic voting schemes are described below.

Accountability of anonymous channel/mixer schemes. On the surface, anonymous channel/mixer schemes appear to meet audit requirements because there is a verifiable public record of the registered voters who asked for a signed, serialized,

blinded ballot. Most likely, this would take the form of a public database containing all of the voter-signed ballot requests from the first half of the protocol. This prevents the authority from generating extra fake ballots or deleting valid ballots, since the number of ballots must match the number of valid requests. Since each request must be signed with a registered voter's private key, the authority cannot mask extra ballots by submitting and fulfilling fake ballot requests for registered voters who did not vote. When the polls close, an authority would make sure that the signature on each request matches one, and only one, registered voter's public key, and that the number of ballot requests matches the number of submitted ballots.

There is a problem in that the number of submitted ballots legitimately can be less than the number of ballot requests. The leading cause for this would be a voter changing his/her mind about voting after receiving a ballot but before submitting it. With a good acknowledge protocol, this would require canceling the vote or turning off the computer in a very narrow time window. It could also happen if a network outage or failure of the voter's computer occurs in the same time window and is not corrected before the polls close. Both cases are fairly unlikely, but they do have a finite probability and the odds improve as the number of voters increases. Authorities would have a tough time tracking down the reason(s) for the discrepancy. Conceivably, a voter could prove a failure by providing a copy of the signed ballot serial number after the polls close.

Authorities would know that the number was issued but not used, and could allow the voter to restart the protocol from the beginning. In any event, the number of missing ballots from this source should be small, which prevents the authority from throwing away many ballots. However, the hole might be more serious for very small precincts or in races decided by a small margin.

The most serious attacks, which are a direct consequence of the lack of verifiability, keep the number of requests and the number of ballots identical. One attack involves the authority (1) modifying (and resigning) submitted ballots or (2) deleting and substituting new ballots. Another attack involves deleting ballots and a corresponding number of random ballot requests from the database.

These attacks are possible because the protocol deliberately severs the connection between the ballot request and the submitted ballot. As such, there is no way

that an independent observer can detect that the ballots are the same as those that were submitted, or even that all submitted ballots are still present.

An appropriate counter to these attacks is a bit commitment on the signed ballot returned to the voter as noted by the papers by A. Fujioka, T. Okamoto, K. Ohta and Park, K. Itoh, K. Kurosawa. The voter, then, can detect attacks where the voted ballot and/or ballot serial number have been modified by checking to make sure that his/her ballot number is present among the published ballots and that it contains the correct vote. Note that the voter must also make sure that his/her signed ballot request is present; otherwise, the authority could have deleted it to offset the deletion of someone else's ballot. If something is amiss, the voter can dispute the election results. Again, this is weak protection because a significant number of voters must perform the verification.

It might be argued that a corrupt authority would have to gamble that most voters will not bother to verify their requests and ballots. The larger the number of ballots modified or deleted, the greater the risk of exposure. Even if there is no way to prove that the disputing voters are not lying, a proportionately large number of disputes would certainly raise suspicion and lend credibility to the disputers. In any event, no concrete proof of tampering may be possible. In the end, anonymous channel/mixer schemes completely rely on voter verification to ensure that there has been no tampering.

Traffic analysis of anonymous channel/mixer schemes. With anonymous channel/mixer schemes, concealing the network address of a submitted ballot is not good enough for very slow traffic periods. For example, if only one voter submits a ballot for signature and the corresponding unblinded ballot in a given period, the authority will know how the voter voted, even if the network address is concealed. A cumbersome remedy might be to buffer ballots on some machine under the control of a different set of authorities.

Multi-way homomorphic schemes. Previously homomorphic schemes exhibit universal verifiability for simple ballots (*i.e.*, *yes/no*, *N*-of-*N* approval) but do not generalize to *d*-of-*N* multi-way ballots. The system under Cramer et al., contains a multi-way ballot methodology, but it only works for ballots in which the voter must choose 1-of-*N* candidates. No previous systems address ballots in which the voter may choose *d*-of-*N* candidates. Such ballots are quite common in both public and private elections. Also, the Cramer et al., system employs an algorithm for determining the tally as $O(l^{N-1})$,

which is not practical for commonly encountered values of N and/or large numbers of voters, l. Thus, this system may be practical for *yes/no* elections, but may not be practical for multi-candidate races containing more than a few candidates in large precincts.

SUMMARY

The election scheme presented here overcomes these limitations by (1) significantly reducing the work required to tally a multi-way election and (2) enforcing that multi-way ballots are well-formed without opening the ballots. We also present our scheme in both Z_p, elliptic curve and general group contexts, present techniques for supporting write-in candidates, and provide a practical implementation approach for supporting an uneraseable "ballot box."

Efficient multi-way ballots. Described below is an efficient multi-way scheme and associated protocol that reduces the work in tallying the election from $O(l^{N-1})$ to $O(lN)$. This improvement is a direct outgrowth of converting a d-of-N option into N, 1-of-2 options and tallying the votes for each option separately. Only N-1 (instead of N), $O(lN)$ tallies must be computed for d-of-N races.

Multi-way zero-knowledge proofs. The above ballot formulation alone is insufficient to restrict voting for more than d options (*i.e.*, submitting an ill-formed ballot). To guard against such attacks, the N encrypted votes are subjected to a zero-knowledge proof of validity. The proof ensures that the encrypted ballot meets the d-of-N condition (*i.e.*, no overvoting) without revealing which d options are selected.

Elliptic curve and General Group formulations. Beyond the Z_p setting, the scheme is extended to the elliptic curve setting and more general group settings. This includes adapting threshold cryptosystem protocols to elliptic curves and developing efficient zero-knowledge proofs of validity.

Support for write-in candidates. No work has been done to support the seldom used, but frequently required, write-in candidate. The scheme provides a write-in framework that integrates well within the protocol.

Although the papers by J. Benaloh, M. Yung and Cramer, R. Gennaro, B. Schoenmakers make the observation that multi-way races can be reduced to independent races, the prior art fails to address the problem of how to limit the number of choices,

which is normally required. The below multi-way zero knowledge proof does so by combining the individual *yes/no* votes into a single zero-knowledge proof. The proof ensures that the voter casts no more than *d yes* votes.

"Amortization," in cryptography, can be used to combine multiple votes into a single zero-knowledge proof, thus saving significant bandwidth and computation. However, the prior art, namely the paper by K. Sako, J. Kilian, does not discuss the ability of the proof to limit the number of *yes* votes. Furthermore, proofs under the prior art, namely the paper by R. Cramer, R. Gennaro, B. Schoenmakers, only ensures that the votes are well-formed, and rely on the tally computation to limit the voter to a single yes vote. Prior approaches apply amortization to the specific problem of proving *d*-of-*N* ballots.

The scheme below effectively characterizes multi-way races as being reduced to *N* individual *yes/no* races. Thus, the scheme described below employs homomorphic encryption to reduce a balance of *N* candidates to *N* number of binary or *yes/no* votes, one for each candidate. Thus for a ballot of six candidates, a voter votes for one of six, and no for the other five. The inventive protocol ensures that each voter indeed voted for only one of six candidates, under this example. Benaloh and Yung state that approval voting (where a voter may cast a vote for any number of the given candidates) is really just a case of several independent *yes/no* votes. However, although the observation is made, no method is given for processing multi-way elections that reduces to several independent *yes/no* votes and enforces well-formed, *d*-of-*N*, votes.

The computational load of prior systems is significantly reduced from exponential to linear. Our methodology supports ballots where the voter may choose *d*-of-*N* candidates, which effectively handles both the 1-of-*N* case and the, more general, *d*-of-*N* case. Moreover, the computational load required to determine the tally is linear instead of exponential in *N* as under the method described by R. Cramer, R. Gennaro, B. Schoenmakers, so it can practically support large values of *d*, *N*, and *l*. Table 1 illustrates the comparative computational load improvement.

Table 1: Computational requirements comparison of technique by R. Cramer, R. Gennaro and B. Schoenmakers vs. the present scheme

Precinct Voters, l	Candidates, N	Prior Art	Present Scheme
10	5	10^4	40
100	5	10^8	400
1000	5	10^{12}	4000
10	10	10^9	90
100	10	10^{18}	900
1000	10	10^{27}	9000

Described below is a formulation for using elliptic curves for both the zero-knowledge proofs and the tallying, which improves security, reduces the required key lengths dramatically, and significantly reduces bandwidth requirements. Both the systems by K. Sako, J. Kilian, and R. Cramer, R. Gennaro, B. Schoenmakers, mention using elliptic curves instead of subgroups of Z_p. Neither presents a methodology for doing so, and neither discusses the significant efficiency enhancements of elliptic curves within the voting context. Furthermore, both contexts are adapted to the below scheme for efficient multi-way elections.

In sum, the invention described herein overcomes the limitations of the prior art and provides numerous additional benefits. The above brief summary presents only some embodiments or aspects of the invention. Some simplifications and omissions were made in the summary, because the summary is intended to highlight and introduce only some aspects of the disclosed embodiments, not to limit the scope of the invention. Presented below is a detailed description of illustrated and suitable embodiments, which will permit one skilled in the relevant art to make and use aspects of the invention. One skilled in the relevant art can obtain a full appreciation of aspects of the invention from the detailed description, read together with the figures, and from the claims (which follow the detailed description).

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating a suitable environment for implementing embodiments of the invention.

Figure 2 is flow diagram illustrating voter ballot creation voting protocol under one embodiment.

Figure 3 is a flow diagram illustrating a ballot verification protocol under one embodiment.

Figure 4 is a flow diagram illustrating a complete multi-way electronic election protocol.

Figure 5 is a flow diagram showing a tabulation search enhancement for more rapidly computing a decrypted tally of votes, over trial and error methods.

In the drawings, identical reference numbers identify identical or substantially similar elements or steps. To easily identify the discussion of any particular element or step, the most significant digit or digits in a reference number refer to the figure number in which that element or step is first introduced and discussed (*e.g.*, step 204 is first introduced and discussed with respect to Figure 2). The headings provided herein are for convenience only, and do not affect the scope or meaning of the claimed invention.

DETAILED DESCRIPTION

The following description provides specific details for a thorough understanding of, and enabling description for, embodiments of the invention. However, one skilled in the relevant art will understand that the invention may be practiced without these details. In other instances, well-known structures and functions have not been shown or described in detail to avoid unnecessarily obscuring embodiments of the invention.

Much of the detailed description provided herein is explicitly disclosed in the provisional patent applications noted above; most or all of the additional material will be recognized by those skilled in the relevant art as being inherent in the detailed description provided in such provisional patent applications, or well known to those

skilled in the relevant art. Those skilled in the relevant art can implement the invention based on the detailed description provided in the provisional patent applications.

The system is described below with l voters $V_1,\ldots,V_l$ and n authorities $A_1,\ldots,A_n$ interacting through a "bulletin board," typically implemented by a web server. Ballots are posted by voters to an area on the bulletin board called a "ballot box."

Ballots are encrypted and are never decrypted. The scheme uses Z_p, elliptic curve or general group cryptosystems. Homomorphic properties of the encryption allow ballots to be "combined" to produce encrypted tallies. This ensures "universal verifiability" since anyone can (1) see *who* voted without seeing *how* they voted, (2) verify that all ballots are well formed, (3) duplicate the combination of the encrypted ballots to obtain the encrypted tally, and (4) verify computation of the decrypted tally from the encrypted tally.

Ballots are accompanied by zero-knowledge proofs of validity to insure that a voted ballot includes allowable options, without leaking any information about which ballot option is chosen. Furthermore, the below system employs the Fiat-Shamir heuristic, which is described in A. Fiat, A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," Advances in Cryptology - CRYPTO '86 Lecture Notes in Computer Science, pp. 186-194, Springer-Verlag, New York, 1987, to create non-interactive proofs.

The encrypted tally is decrypted by t-of-n authorities without reconstructing the authorities' private keys, using threshold encryption techniques. The decryption protocol requires a zero-knowledge proof which ensures that the correct ciphertext has been decrypted using the private-key share corresponding to the authorities' group public-key. Hence, compromise of the election results or voter privacy would require a conspiracy of at least t-of-n authorities. Even if all authorities conspire, they cannot conspire to fabricate the election results.

The mathematical notation used here is readily understandable to those skilled in the relevant art; however, for those unfamiliar with the art, the following definitions and descriptions are provided. Such definitions, although brief, will help those generally unfamiliar with the art to more fully understand aspects of the invention based on the detailed description provided herein. Such definitions are further defined by

the description of the invention as a whole (including the claims), and not simply by such definitions.

Figures 2, 3, 4 and 5, as well as the detailed description provided herein, describe protocols between a voter and verifier (or between a proving party and a verifying party). The action performed by the parties are grouped together into flow diagram boxes. Each line of text or equations in a box describes a step, such as a computation, transmittal of information, or storage or retrieval function. Such flow diagrams are read line by line and box by box.

The term "party" as generally used herein, indicates an entity, and might be an agent who performs a step or a collection of steps under the protocol. It may also refer to a means or method for performing some or all of the steps. Thus, some or all portions of the protocols may be performed under any suitable configuration of digital logic circuitry. For example, any or all steps under the protocol may be realized by not only a general purpose computer, such as a personal computer, but by a hard-wired or dedicated combinatorial logic device, or any sort of suitably programmed machine or microprocessor, so long as such device or machine performs the required processing steps, storage, input and output, and the like.

The symbol "$\leftarrow$" generally denotes the assignment function, which indicates that a variable or symbol on a left side of an equation is assigned a value or values on a right side of that equation. Assignments do not necessarily imply storage space or memory use, but may refer to intermediate computations performed in random-access memory or within a CPU.

The symbol "$\in_R$" generally indicates that a number or numbers on the left-hand side are chosen from a set on the right-hand side according to a probability distribution that is substantially uniform and independent (random). In practice, a physical random number generator can be used, possibly in conjunction with additional post-processing, or a deterministic pseudo-random number generator. Methods of generating random numbers are known by those skilled in the relevant art.

The symbol "Z_p" denotes a set of numbers of integers 0 through P-1, or ring of integers, modulo P. Addition and multiplication of elements in the ring Z_p are defined modulo P.

The symbols "Π" and "Σ" respectively denote product and sum, which are indexed.

Figure 1 and the following discussion provide a brief, general description of a suitable computing environment in which aspects of the invention can be implemented. Although not required, embodiments of the invention will be described in the general context of computer-executable instructions, such as routines executed by a general-purpose computer, such as a personal computer or web server. Those skilled in the relevant art will appreciate that aspects of the invention (such as for small elections) can be practiced with other computer system configurations, including Internet appliances, hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, mini computers, cell phones, mainframe computers, and the like. Aspects of the invention can be embodied in a special purpose computer or data processor that is specifically programmed, configured or constructed to perform one or more of the computer-executable instructions explained herein. Indeed, the term "computer," as generally used herein, refers to any of the above devices, as well as any data processor.

The invention can also be practiced in distributed computing environments where tasks or modules are performed by remote processing devices, which are linked through a communications network, such as a Local Area Network (LAN), Wide Area Network (WAN), and the Internet. In a distributed computing environment, program modules or sub-routines may be located in both local and remote memory storage devices. The invention described herein may be stored or distributed on computer-readable media, including magnetic and optically readable and removable computer disks, as well as distributed electronically over the Internet or other networks (including wireless networks). Those skilled in the relevant art will recognize that portions of the protocols described herein reside on a server computer, while corresponding portions reside on client computers. Data structures and transmission of data particular to such protocols are also encompassed within the scope of the invention.

Unless described otherwise, the construction and operation of the various blocks shown in Figure 1 are of conventional design. As a result, such blocks need not be described in further detail herein, as they will be readily understood by those skilled in the relevant art.

Referring to Figure 1, a suitable environment of system 100 includes one or more voter or client computers 102, each of which includes a browser program module 104 that permits the computer to access and exchange data with the Internet, including web sites within the World Wide Web portion 106 of the Internet. The voter computers 102 may include one or more central processing units or other logic processing circuitry, memory, input devices (*e.g.*, keyboards and pointing devices), output devices (*e.g.*, display devices and printers), and storage devices (*e.g.*, fixed, floppy, and optical disk drives), all well known but not shown in Figure 1. The voter computers 102 may also include other program modules, such as an operating system, one or more application programs (*e.g.*, word processing or spread sheet applications), and the like. As shown in Figure 1, there are *N* number of voter computers 102, representing voters 1, 2, 3 . . . *N*.

A server computer system 108 or "vote collection center," coupled to the Internet or World Wide Web ("Web") 106, performs much or all of the ballot collection, storing and other processes. A database 110, coupled to the server computer 108, stores much of the web pages and data (including ballots) exchanged between the voter computers 102, one or more voting poll computers 112 and the server computer 108. The voting poll computer 112 is a personal computer, server computer, mini-computer, or the like, positioned at a public voting location to permit members of the public, or voters who may not have ready access to computers coupled to the Internet 106, to electronically vote under the system described herein. Thus, the voter computers 102 may be positioned at individual voter's homes, where one or more voting poll computers 112 are located publicly or otherwise accessible to voters in a public election. The voting poll computer 112 may include a local area network (LAN) having one server computer and several client computers or voter terminals coupled thereto via the LAN to thereby permit several voters to vote simultaneously or in parallel.

Under an alternative embodiment, the system 100 may be used in the context of a private election, such as the election of corporate officers or board members. Under this embodiment, the voter computers 102 may be laptops or desktop computers of shareholders, and the voting poll computer 112 can be one or more computers positioned within the company (*e.g.*, in the lobby) performing the election. Thus, shareholders may visit the company to access the voting poll computer 112 to cast their votes.

One or more optional authority or organization computers 114 are also coupled to the server computer system 108 via the Internet 106. The authority computers 114 each hold a key necessary to decrypt the electronic ballots stored in the database 110. Threshold cryptographic systems require that a subset t of the total number of authorities n (i.e., t<n) agree to decrypt the ballots, to thereby avoid the requirement that all authorities are needed for ballot decryption. The authority computers 114 may provide their decryption share to the server computer system 108 after the voting period ends so that the server computer system may decrypt the ballots and tally the results.

The server computer 108 includes a server engine 120, a web page management component 122, a database management component 124, as well as other components not shown. The server engine 120 performs, in addition to standard functionality, the electronic voting protocol. The above encryption protocol may be stored on the server computer, and portions of such protocol also stored on the client computers, together with appropriate constants. Indeed, the above protocol may be stored and distributed on computer readable media, including magnetic and optically readable, and removable computer disks, as well as distributed electronically over the Internet or other networks. Those skilled in the relevant art will recognize that portions of the protocol reside on the server computer, while corresponding portions reside on the client computer. Data structures and transmission of data particular to the above protocol are also encompassed within the present invention. Thus, the server engine 120 may perform all necessary ballot transmission to authorized voters, ballot collection, verifying ballots (*e.g.*, checking digital signatures and passing verification of included proofs of validity in ballots), vote aggregation, ballot decryption and/or vote tabulation. Under an alternative embodiment, the server engine 120 simply collects all electronic ballots as a data collection center. The electronic ballots are then stored and provided to a third party organization conducting the election, such as a municipality, together with tools to decrypt the tally and produce election results.

The web page component 122 handles creation and display or routing of web pages such as an electronic ballot box web page, as described below. Voters and users may access the server computer 108 by means of a URL associated therewith, such as http:\\www.votehere.net, or a URL associated with the election, such as a URL for a municipality. The municipality may host or operate the server computer system 108

directly, or automatically forward such received electronic ballots to a third party vote authorizer who may operate the server computer system. The URL, or any link or address noted herein, can be any resource locator.

The web page management process 122 and server computer 108 may have secure sections or pages that may only be accessed by authorized people, such as authorized voters or system administrators. The server computer 108 may employ a secure socket layer ("SSL") and tokens or cookies to authenticate such users. Indeed, for small elections, or those where the probability of fraud is low (or results of fraud are relatively inconsequential), the system 100 may employ such simple network security measures for gathering and storing votes as explained below, rather than employing complex electronic encrypted ballots, as described in the above-noted patent application. Methods of authenticating users (such as through the use of passwords), establishing secure transmission connections, and providing secure servers and web pages are known to those skilled in the relevant art.

The election scheme and system uses a "bulletin board" where each posting is digitally signed and nothing can be erased. See papers by K. Sako, J. Kilian, R. and Cramer, R. Gennaro, B. Schoenmakers. The bulletin board is implemented as a web server. The "ballot box" resides on the bulletin board and holds all of the encrypted ballots. Erasing can be prevented by writing the web server data to a write-once, read-many (WORM) permanent storage medium or similar device. Further details on such a bulletin board system are found in U.S. Patent Application No. _____, filed March 24, 2000, entitled Electronic Voting Scheme Employing Permanent Ballot Storage, assigned to the assignee of the present invention.

Two embodiments of the system are described herein, which respectively employ a Z_p (ElGamal) cryptosystem, an elliptic curve cryptosystem and cryptosystem under general groups. Both cryptosystems employ public keys for asymmetrical cryptosystems. Public key systems employ so-called one-way functions and trap-door functions. A "one-way function" is a function that is relatively easy to calculate output therefrom but whose inverse functions are far more difficult to compute. For example, power functions are such that they are easy to compute the product of a number of equal factors, but the reverse operation of finding the root of a quantity, is more complicated. "Trap door" functions are similar to one-way functions, but where the inverse functions

are extremely difficult unless some additional information is available. This additional information is typically the "private key" held by a party, such as a voter.

ElGamal cryptosystem. The ElGamal cryptosystem introduced in W. Diffie, M. E. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, 22(6):644654, 1976; and improved on in T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, IT-31(4):469-472, 1985, is implemented for unique subgroups of G_q of order q of Z^*_p, where p and q are large primes such that q divides p-1 (*i.e.*, $q|p$-1). The primes, p and q, and generator g of G_q are system parameters. The private key is designated as s, which is selected at random from the group Z_p, *i.e.*, $s \in_R Z_p$, with corresponding public key h being generated by: $h = g^s$. All calculations are modulo p unless otherwise noted.

To encrypt a plaintext message m, the message is generated based on the group G_q, *i.e.*, $m \in G_q$, the sender party chooses a random number, $\alpha \in_R Z_q$, and sends the ciphertext $(x,y) = (g^\alpha, h^\alpha m)$ to the receiver. To decrypt the ciphertext, the receiver party uses their private key, s, and recovers the plaintext as $m = y/x^s$.

Elliptic curve cryptosystem. The ElGamal cryptosystem can be implemented in elliptic curve as noted in N. Koblitz, "A Course in Number Theory and Cryptography," 2nd edition, Springer, 1994. Encipherment and decipherment under the elliptic curve embodiment may require less storage than that under the Z_p embodiments. A publicly known finite field, say Z_p, elliptic curve E defined over Z_p, and a base point $B \in E$ are defined as system parameters. The private key is $s \in_R Z_p$ with corresponding public key $h = sB$.

To encrypt a plaintext message $m \in Z_p$, the sender chooses a random number, $\alpha \in_R Z_p$, and sends the ciphertext $(x,y) = (\alpha B, G_m + \alpha h)$ to the receiver; $G_m \in E$ is the encoding of m onto the elliptic curve E. To decrypt the ciphertext, the receiver uses their private key, s, and recovers the plaintext as $G_m = y - sx = (G_m + \alpha h) - s(\alpha B)$.

1. Proofs of Knowledge

Because the system does not decrypt individual voted ballots during election tallying, the system must ensure that all ballots contain only allowable choices. This assurance can not be at the expense of voter privacy. The system employs three-move, zero-knowledge proofs where the voters supply a proof of validity alongside their

ballots. This proof of validity verifies that the encrypted ballot contains only allowable choices without disclosing information about which ballot choices were made.

"Zero-knowledge proofs" allow a voter or prover party to demonstrate knowledge of a secret while revealing no information whatsoever of use to the verifier party in conveying this demonstration of knowledge. Only a single bit of information is conveyed, namely that the prover party actually does know the secret. In other words, a voter and a verifying authority exchange messages, where the voter's objective is to convince the verifier the truth of an assertion, *e.g.*, that the encrypted ballot is complete and correct, without revealing how the voter voted on each element in the ballot. Under such zero-knowledge proofs, each voter effectively generates certain numbers that only a person having his or her own private key could generate. A verifier or authenticating party then confirms that each calculated number indeed is generated under a known algorithm to thereby authenticate the voter and that his or her electronic ballot is complete and correct or "well-formed" for all allowable choices and constraints.

In order to make the protocol non-interactive, the system uses aspects of the Fiat-Shamir heuristic, which incorporates a hash function, H. The Fiat-Shamir heuristic allows conversion of the three-move, interactive protocol into a single, non-interactive data or ballot submission consisting of a (1) a *statement*, (2) a *challenge* (which is a hash of the statement), and (3) a *response*. Furthermore, in order to prevent vote-duplication, each challenge is made voter specific

$$H(h_i,x,y,a_1 \ldots a_N,b_1 \ldots b_N), \tag{1}$$

where h_i is the public-key for voter V_i. See, *e.g.*, N. Koblitz, above. The hash function H is effectively a public one-way function for checking message or ballot integrity. A hash function is a special function in the sense that it maps a set of input values of variable length and value onto a set of output values of a constant length. The hash function is publicized so that anyone can calculate the hash result and verify a received message. Many known hash functions may be employed, and may be collision-free, in that it is effectively impossible to construct two different messages with the same resulting hash code.

In the system's multi-way ballot scheme, a *d*-of-*N* option is converted instead into *N*, 1-of-2 options (*i.e., yes/no*) and votes are accumulated for each option in parallel. Furthermore, in order to ensure that the ballot is well formed, the system enforces the *d*-of-*N* restriction in zero-knowledge.

Proof of knowledge for discrete logs. This section describes proofs of knowledge for discrete logs using aspects of the techniques from the papers by R. Cramer, R. Gennaro, B. Schoenmakers and R. Cramer, I. Damgrd, B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," Advances in Cryptology- CRYPTO '94, Lecture Notes in Computer Science, pp. 174-187, Springer-Verlag, Berlin, 1994.

Consider a *d*-of-*N* vote, which consists of n ElGamal encryptions of the following form:

$$(x_i, y_i) = (g^{\alpha_i}, h^{\alpha_i} G^{v_i}), \text{ with } v_i \in \{-1, 1\},$$

where the prover knows the values of α_i and v_i. (One skilled in the art can observe that the values of 1 and -1 are not essential; other non-equal pairs may be used. For an example of this, see section 4.3 below.) To show that exactly *d* of the *n* v_is are equal to 1 and *n-d* of them are equal to -1, the prover first proves that at least *d* of the v_is are 1, then proves that at least *n-d* of them are -1. The first part consist of a witness indistinguishable proof of knowledge of the relation that at least *d* of the following equalities are satisfied:

$$\log_g x_i = \log_h y_i / G$$

(The second part is similar and will be omitted from this description.)

This proof is constructed using aspects of the techniques noted in the paper by R. Cramer, I. Damgrd, B. Schoenmakers, by combining the proof of knowledge by Chaum and Pedersen and the secret sharing scheme by Shamir. By Lemma 1 noted in the paper by R. Cramer, R. Gennaro, B. Schoenmakers, the Chaum-Pedersen protocol satisfies the conditions for the construction noted in the paper by R. Cramer, I. Damgrd, B. Schoenmakers.

The proof consists of n Chaum-Pedersen proofs where the challenges are considered shares in Shamir's secret sharing scheme. The protocol starts with the generation of *n-d* simulated Chaum-Pedersen proofs for $\log_g x_i = \log_h y_i / G$, for *i* where $v_i \neq 1$. The challenges s_i in these simulated proofs are selected at random by the prover

party or voter. The prover party then computes the first pass (a_i,b_i) for the rest of the Chaum-Pedersen proofs where $v_i = 1$ (also called real proofs). Under one embodiment explained below, all (x_i,y_i,a_i,b_i) are sent to the verifier party in return for a challenge c. (In the non-interactive version they are hashed along with a unique public string identifying the voter to form c.)

Shamir's secret sharing scheme is then used to determine the challenges for the real proofs. This is done by interpolating a polynomial P over Z_q with degree $(n\text{-}d)$ or less that satisfies $P(0) = c$ and $P(i) = s_i$ for each i such that $v_i \neq 1$, and setting each real challenge s_i to $P(i)$. Finally the real proofs are completed. The prover party sends the values x_i, y_i, a_i, b_i, s_i, and r_i. Under an alternative embodiment, s_i is not sent; instead, the coefficients of the polynomial P are sent, as explained below.

Verification is done by checking each Chaum–Pedersen proof separately and then verifying that there exists a polynomial P over Z_q with degree $(n - d)$ or less such that $P(0) = c$ and $P(I) = s_i$. Under the alternative embodiment noted above, the interpolated polynomial is explicitly available in the proof. The verifier party just checks that the values of it at corresponding valuation points are correct. Figures 2 and 3 show the complete ballot validity proof and verification as they are used in the election protocol.

Referring to Figure 2, a ballot creation protocol or routine 200 is shown that is typically performed by the voter computer 102. In step 202, the voter computer 102 generates a set of constants for each of number i ballot initiatives, where each choice on the ballot is reduced to a single yes or no vote (*i.e.*, $v_i = 1$ or = -1, respectively). Step 202 (as well as steps 206 and 210) are divided by a vertical line, whereby the left hand column created by the line represents a yes vote (*i.e.*, = 1), while the right hand column represents a no vote (*i.e.*, =-1), or vice versa. The actual assignment of the yes vote = 1 and a no vote = -1 is arbitrary. As shown in step 202, the voter computer 102 initially selects at random several constants from the ring Z_p, and therefrom generates the constants x, y, a and b, all as shown

In step 204, the voter computer 102 applies a hash function H to all of these generated constants, as well as an identifier value for the voter or digital signature (ID), to generate a hash function result c_1. The voter computer 102 also interpolates a polynomial P over the ring Z_q, such that the degree of the polynomial is less than or equal to the

number of ballot options n minus the number of choices of those options d. The value of $P(o)$ is set as the result of the hash function, while $P(j) = s_{j,1}$ for $v_j = -1$.

In step 206, the voter computer 102 generates the result of the polynomials $s_{j,1}$, and therefrom generates new constants r, a, and b for a yes vote; for a no vote, the voter computer 102 simple generates the constants a and b, all as shown.

In step 208, the voter computer 102 generates a new hash function result c_2, and a new polynomial P_2 that now has degree less than or equal to d. Additionally, $P(0)$ is equal to the new result of the hash function and $P(j)$ is equal to $s_{j,2}$ for a yes vote = 1.

In step 210, the voter computer 102 generates the result of the new polynomial as s and therefrom generates the constant r.

In step 212, the voter computer 102 finally outputs a string of constants in a precise order, as shown in Figure 2.

Referring to Figure 3, a verification protocol or routine 300 is shown that is performed by the server computer system 108. In step 302, the server computer system receives the string of output values, in the exact order, as they were output under step 212 above. In step 304, the server computer performs the public hash function on the string to determine if the result is equal to c_1. If so, the server computer determines whether there exists a polynomial selected from the ring Z_q that has degree $n - d$. The server computer also then computes values a and b, and determines whether such computer values correspond to those received previously under step 302. Under an alternative embodiment, the server computer simply verifies values of the polynomial $P(x)$ at values 1, 2, 3 . . . i and that the values at a and b are correct.

In step 306, the server computer effectively performs the same steps, but for the no vote or -1 portion.

Having described the above ballot formation and verification with respect to Figures 2 and 3, the remaining discussion is readily understandable by those skilled in the relevant art based on the following description as applied to the system of Figure 1. In general, ballot creation and proof of knowledge steps are performed by the voter computer 102, while verification and tallying steps are performed by the server computer system 108. Thus, for reasons of economy and clarity, the following discussion of the voting encryption protocol, including homomorphic encryption, threshold cryptosystems, and a multi-way election scheme are presented generally without direct reference to the

computing platforms of Figure 1. Those skilled in the relevant art will recognize that nearly all elements of the protocols described herein are performed by computers, except for the election preparation or generation of a ballot by an organization and the creation of choice or casting of the ballot by the voter (*i.e.*, the actual human choices made). The term "computer" as used herein refers to any data processing device, as described more fully herein.

The preceding covers the situation where a voter is *required* to vote for *exactly d* of the *n* choices (where *d* represents the number of yes choices and *d* - *n* represents the number of no choices). However, this methodology can be easily extended to a situation where a voter is asked to choose *at most d* from a set of *n* possible choices. To facilitate this, the ballot is constructed to return $n + d$ decisions on the issue, instead of just *n*. By convention, the first *n* of these correspond to the *n* available decisions, while the last *d* of these all correspond to decisions known as "do not care." The do not care decisions are not expected to be tallied, however, when a voter proves that he or she has chosen *exactly d* in this situation, the protocol also proves that he or she has selected *at most d* from the first *n* "actual decisions" in contrast to the last *d* "do not care" decisions. Those skilled in the relevant art will recognize that the same technique may readily extend to situations of the form: *n* choices, select "at least" d_1 *and* "at most" d_2.

Proof of knowledge for elliptic curve. In this section, we describe proofs of knowledge for elliptic curves. We begin describing a public coin protocol for proving knowledge of an integer $\omega \in Z$ such that $\omega B = P$. This is the discrete log problem on E, where E is an elliptic curve over Z_p and $B, P \in E$. Table 2 shows the protocol similar to C. P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, 4(3):161-174, 1991.

Lemma 1. The protocol of Table 2: Proof of knowledge for discrete log problem on elliptic curve E is a three–move, public coin [key?] proof of knowledge for the relation ωB (the discrete log problem on elliptic curve E). The proof satisfies special soundness, and is special verifier honest–verifier zero-knowledge. In general, a proof is "sound" if there exists an expected polynomial-time algorithm, whereby if a dishonest prover impersonating a voter can with non-negligible probability successfully execute the protocol with another, such as the verifying authority, then the algorithm can be used to

extract from this prover knowledge which will, with overwhelming probability, allow successful subsequent protocol executions.

Prover $\alpha \in_R Z_p$		Verifier
$a = B\alpha$	$a \rightarrow$	
	$\leftarrow c$	$c \in_R Z_p$
$r = \alpha + c\omega$	$r \rightarrow$	$rB = a + Pc$

Table 2: Proof of knowledge for discrete log problem on elliptic curve E

Proof. It is easily shown that

$$rB = a + Pc$$
$$(\alpha + c\omega)B = B\alpha + (\omega B)c$$

Special soundness holds because from two accepting conversations with the same first move $(a,b,c_1;r_1)$, $(a,b,c_2;r_2)$ and $c_1 \neq c_2$; a witness $\frac{\cdot}{c_1 - c_2}$ can be extracted satisfying $(x,y) = (\omega B, \omega h)$. Honest–verifier zero-knowledge holds because, for random c and r, we have $(Br - xc,\ hr - yc,c,r)$ is an accepting conversation with the right distribution. Since the challenge c can be chosen freely, we also have special honest–verifier zero-knowledge.

2. Homomorphic Encryption

The homomorphic property is important for meeting universal verifiability requirements in concert with threshold cryptosystems. We give a definition of this property here.

Definition 1: Let G be a group, and let S and H be sets. A *keyed encryption*

$$E: G \times S \rightarrow H$$

function on G is a function

Definition 2: With the notation above, if H is a group (i.e. a set with arithmetic structure), and if the function E satisfies

$$\forall m_1, m_2 \in G, \forall \alpha \in S \quad \exists \alpha_1, \alpha_2 \in S \quad \text{such that} \quad E(m_1 m_2, \alpha) = E(m_1, \alpha_1)E(m_2, \alpha_2)$$

then E is a *homomorphic encryption function.*

Definition 3: If S is also a group and if the stronger condition

$$\forall m_1, m_2 \in G, \forall \alpha_1, \alpha_2 \in S \quad E(m_1 m_2, \alpha_1 \alpha_2) = E(m_1, \alpha_1)E(m_2, \alpha_2)$$

then E is a *strongly homomorphic encryption function*

Note: Both definition 2 and definition 3 can be easily rephrased for groups whose binary operation is written in additive notation (usually the case for commutative groups). For example, if S and G are both written with additive notation, the equation of definition 3 becomes

$$\forall m_1, m_2 \in G, \forall \alpha_1, \alpha_2 \in S \quad E(m_1 + m_2, \alpha_1 + \alpha_2) = E(m_1, \alpha_1) E(m_2, \alpha_2)$$

(here H is still using multiplicative notation).

ElGamal homomorphism. For the Z_p ElGamal cryptosystem, as noted above, given a fixed generator $G \in G_q$, the encryption of a message $m \in Z_q$ is

$$E(m, \alpha) = (g^{\alpha}, h^{\alpha} G^{m})$$

Hence, in definition 2, the group, H, can be taken to be $Z_p \times Z_p$ with coordinate wise multiplication, and we have

(2)

$$E(m_1 + m_2, \alpha_1 + \alpha_2) = (g^{\alpha_1 + \alpha_2}, h^{\alpha_1 + \alpha_2} G^{m_1 + m_2}) = (g^{\alpha_1}, h^{\alpha_1} G^{m_1})(g^{\alpha_1}, h^{\alpha_1} G^{m_1})$$

$$= E(m_1, \alpha_1) E(m_2, \alpha_2)$$

So the ElGamal cryptosystem is *strongly homomorphic.*

Further, if $h = g^s$, and if $(X, Y) = E(m, \alpha) = (g^{\alpha}, h^{\alpha} G^{m})$ is the encryption of some message, then we always have

$$W = y / x^s = (h^{\alpha} G^{m}) / (g^{\alpha})^s = G^{m}$$

Computation of the exponent, m, from W requires computation of a discrete log, which is difficult in general. However, because of the relatively small number of votes that will occur in application, m will be relatively small, and this discrete log can computed by simple trial and error. (Here anything less than about 10^{12} is relatively small.)

Elliptic curve homomorphism. Elliptic curve ElGamal cryptosystems also exhibit the required homomorphic property for the voting scheme. This can be verified exactly as in the Z_p case by using $H = \Gamma \times \Gamma$ for a chosen elliptic curve Γ, and by using additive notation for the binary group operation of H instead of multiplicative notation which was used above. Comments regarding the computation of the discrete log apply again, modified only by changing multiplicative notation to additive notation.

3. Threshold Cryptosystems

The objective of a threshold cryptosystem is to share a private key, s, among n members of a group such that messages can be decrypted when a substantial subset, T, cooperate - a (t, n) threshold cryptosystem. Protocols are defined to (1) generate keys jointly among the group, and (2) decrypt messages without reconstructing the private key. T. Pedersen, "A threshold cryptosystem without a trusted party," Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science, pp. 522-526, Springer-Verlag, 1991, describes both protocols for the ElGamal cryptosystem. This protocol extends that to the elliptic curve context.

ElGamal threshold cryptosystem. Solutions to both the key generation and decryption protocols are given in the above reference.

The key generation protocol results in each authority A_j possessing a share $s_j \in Z_q$ of a private key, s. The authorities A commit to these shares by posting the public key, $h_j = G^{s_j}$. Using Shamir's (t, n)-threshold scheme, noted in A. Shamir, "How to share a secret," Communications of the ACM, 22(11):612-613, 1979, the private key can be recovered using Lagrange coefficients without any single authority learning the private key:

$$s = \sum_{j=1}^{t} s_j \lambda_j, \text{ where } \lambda_j = \prod_{k=1, k \in \Lambda, k \neq j}^{t} \frac{k}{k-j} \tag{3}$$

Proof. Evaluate the Lagrange interpolation for polynomials at $f(0)$ and solve for $f_0 = s$.

$$f(z) = \sum_{j=1}^{t} s_{i,j} \prod_{k=1, k \neq j}^{t} \frac{z - z_{i,k}}{z_{i,j} - z_{i,k}} \tag{4}$$

where $f(z) = f_0 + f_1 z + \cdots + f_{t-1} z^{t-1}$.

By substituting $z = z_{i,j}$, all terms in the summation vanish except for the j^{th} term, which is $s_{i,j}$.

As described above, t authorities must cooperate to decrypt ciphertext $(x,y) = (g^{\alpha}, h^{\alpha} m)$ without reconstructing the private key, s:

1. Each authority A_j broadcasts $\omega_j = x^{s_j}$ and proves in zero-knowledge that

$$\log_g h_j = \log_x \omega_j. \tag{5}$$

2. Let Λ denote any subset of t authorities who passed the zero-knowledge proof. Solve for the plaintext; m

$$m = y / \prod_{j=1, j\in\Lambda}^{t} \omega_j^{\lambda} j. \tag{6}$$

Proof.

$$x^s = x^{\sum_{j=1, j\in\Lambda}^{t} s_j \lambda_j} = \prod_{j=1, j\in\Lambda}^{t} x^{s_j \lambda_j} = \prod_{j=1, j\in\Lambda}^{t} \omega^{\lambda} j.$$

$$m = y / x^s = y / \prod_{j=1, j\in\Lambda}^{t} \omega^{\lambda_j}.$$

Elliptic curve threshold cryptosystem. We now adapt the above-noted protocols to the elliptic curve context. The key selection method is as follows:

1. The authorities agree on system parameters: a finite field Z_p, an elliptic curve E defined over Z_p, and a base point $B \in E$.

2. Authority, A_j, chooses private key $s_j \in_R Z_p$ and computes $h_j = s_j B$. Let $C(m, r)$ denote a bit commitment to $m \in 0, 1$ using random string r. A random string, r_j, is chosen and $C_j = C(h_j; r_j)$ is broadcast to all authorities.

3. When all authorities have broadcast their commitment, each A_j opens C_j.

4. The public key, h, is computed as

$$h = \prod_{j=1}^{n} h_j, \text{ where } s = \prod_{j=1}^{n} s_j \text{ is the private key.}$$

Proof.

$$h = \prod_{j=1}^{n} h_j = s_1 B + \cdots + s_n B = B \prod_{j=1}^{n} s_j.$$

After keys have been selected, the server computer system 108 distributes them to the authority computers 114 according to the following protocol:

1. Authority, A_j, chooses a random polynomial $f_j(z) \in F_q$ of degree no greater than $t - 1$ such that $f_j(0) = s_j$. Let

$$f_j(z) = f_{j,0} + f_{j,1} z^{t-1}, \text{ where } f_j(0) = s_j. \tag{7}$$

2. A_j computes $F_{j,i} = B f_{j,i}$, $j = 0, \ldots, t - 1$ and broadcasts $F_{j,i}$, $j = 0, \ldots, t - 1$. Note that $F_{j,0} = h_j$ is known beforehand through the key selection protocol (*i.e.*, $F_j, 0 = B f_{j,0} = B s_j = h_j$.

3. When all authorities have sent these $t-1$ value; A_j sends $x_{j,i} = f_j(i)$ secretly and a signature on $x_{j,i}$ to A_k, $k = 1, \ldots, n$ (A_k keeps x_{kk}).

4. A_j verifies that the share, x_{ij}, received from A_i is consistent with previously published values by verifying

$$Bx_{i,j} = \prod_{l=0}^{t-1} F_{i,l} j^l. \tag{8}$$

Proof.

$$\begin{aligned} B_{i,j} &= B\left(f_{i,0} + f_{i,1} j + \cdots + f_{i,t-1} j^{t-1}\right), \text{since } f_i(j) = x_{i,j} \text{ and} \\ \prod_{l=0}^{t-1} F_{i,l} j^l &= F_{i,0} + F_{i,1} j + \cdots + F_{i,t-1} j^{t-1} \\ &= Bf_{i,0} + Bf_{i,1} j + \cdots + Bf_{i,t-1} j^{t-1}, \text{ since } F_{i,j} = Bf_{i,j} \\ &= B\left(f_{i,0} + f_{i,1} j + \cdots + f_{i,t-1} j^{t-1}\right). \end{aligned}$$

If the verification fails, A_j broadcasts that an error occurred, publishes $x_{j,i}$, the signature, and stops.

5. Authority A_j computes share s_j as the sum of all shares received in Step 3, $\sum_{l=1}^{n} s_{i,j}$. A_j signs public key h.

6. When all authorities have signed public key h, a key authentication center verifies the signatures and makes a certificate showing that h is the group public key for the authority group.

Extending the above to elliptic curves, t authorities must cooperate to decrypt ciphertext $(x, y) = (\alpha B, m + \alpha h)$ without reconstructing the private key, s:

1. Each authority A_j broadcasts $\omega_j = xs_j$ and proves in zero-knowledge that

$$xh_j = B\omega_j \tag{9}$$

Proof.

$$\begin{aligned} xh_j &= B\omega_j \\ x\left(Bs_j\right) &= B\left(xs_{ij}\right) \\ Bxs_j &= Bxs_j. \end{aligned}$$

2. Let Λ denote any subset of t authorities who passed the zero-knowledge proof. Solve for the plaintext, m

$$m = y - \sum_{j=1, j\in\Lambda}^{t} w_j \lambda_j. \tag{10}$$

Proof.

$$\begin{aligned} m &= y - sx \\ &= y - \left(y - \sum_{j=1, j\in\Lambda}^{t} w_j \lambda_j \right) x = y - \sum_{j=1, j\in\Lambda}^{t} x s_j \lambda_j \\ &= y - \sum_{j=1, j\in\Lambda}^{t} w_j \lambda_j. \end{aligned}$$

4. Multi-Way Election Scheme

For support of several options per race, the voting protocol converts the d–of–N option into N, 1–of–2 options and accumulate votes for each option separately. Furthermore, to ensure that the ballot is well-formed (*i.e.*, conforms to the d–of–N restriction), a non–interactive proof is submitted with the ballot. This scheme increases the ballot size by $2(d + N)$ and the proof of validity by N. The voter submits an encrypted ballot to the ballot box accompanied by a zero-knowledge proof as explained above that verifies that the ballot is well-formed (*i.e.*, only allowable choices). Votes are combined using homomorphic techniques as explained above into an encrypted tally. Using threshold encryption coordination by t-of-n authorities as also explained above, the encrypted tally is decrypted without disclosing of the private key. Finally, the tally is computed from the decrypted tally.

4.1 Z_P Formulation

We now describe the election scheme in a discrete log context for subgroups, G_q,of order q of Z_p^*, where p and q are large primes such that q divides $p - 1$ (*i.e.*, $q \mid p-1$). The security of the scheme depends on the security of the ElGamal cryptosystem where the discrete logarithm is considered intractable. Those skilled in the relevant art will appreciate that p and q need not always be prime (but are often prime in

many practical applications). Nevertheless, the discussion herein employs prime values for p and q.

Election preparation. To generate a ballot for an election, the system parameters p, q, and fixed generators g, $G_k \in G_q$ where $1 \le k \le N$ ballot options, are generated jointly by a subset of the authorities. Two more parameters y_k, n_k, Z_p^* are also chosen for each ballot option to represent the decisions 'yes' (y_k) and 'no' (n_k). (In order to simplify implementation, the N generators, G_k, as well as the N pairs y_k, n_k, may be chosen so that they are independent of k. That is, G_k=G, $y_k = y_0$ and $n_k = n_0$, for all $1 \le k \le N$. In fact, from a practical standpoint, it is most convenient to always use the values G_k=$G = g$, and $y_k = y_0 = 1$ and $n_k = n_0 = -1$ or $y_k = y_0 = 1$ and $n_k = n_0 = 0$, but the scheme will work essentially unchanged for any fixed pair of values in Z_P, as long as, of course, $y_0 \neq n_0$. Finally, the authorities cooperate to generate a public key parameter, h, in the cyclic subgroup generated by g, by using the secret sharing protocol described above. Since $h \in \langle g \rangle$, $h = g^s$ for some $s \in Z_p^*$, however, the secret sharing protocol ensures that s is not known by anyone. The authorities may, however, cooperate to compute $y = x^s$ from any given value x without ever revealing information about s.

Casting a ballot. To cast or create a *voted ballot*, an ElGamal encryption is made of each ballot option. For each voted ballot, i, cast in the election, and each ballot option, k, on the ballot, the encrypted message is $G_k^{b(i,k)}$, where G_k is the generator of G_q preassigned to ballot option k, and $b(i,k) \in \{y_k, n_k\}$ is voted ballot i's choice on that ballot option. The encrypted voted ballot, B_i, is then of the form

$$B_i = \{x_k y_k\}_{k=1}^N = \left\{\left(g^{\alpha(i,k)}, h^{\alpha(i,k)} G^{b(i,k)}\right)\right\}_{k=1}^N \quad 1 \le k \le N. \tag{11}$$

where $\alpha(i,k) \in Z_p^*$ is chosen randomly by voter V_i, and kept secret. In addition to the ballot, a non–interactive proof of validity is submitted according to the protocol described above.

Tallying. To tally the election, the votes are combined, exploiting the homomorphic property described above. For discrete logs, this involves multiplying all votes to obtain a single encrypted tally for option k, $(X,Y)_k$. The authorities jointly

execute the decryption protocol described above for $(X,Y)_k$ to obtain $W_k = G_k^{T_k} = Y_k / X_k^s$, where

$$T_k = \sum_{i=1}^{t} b(i,k) = N_k(y)y_k + N_k(n)n_k = N_k(y)y_k + (l - N_k(y))n_k \qquad (12)$$

where $N_k(y)$ and $N_k(n)$ are, respectively, the number of voters, V_i, who chose $b(i,k) = n_k$. (For the convenient parameter choices $y_k = 1$, $n_k = -1$ that will typically be used, T_k is equal to the difference between the number of yes-votes and the number of no-votes for option k, $-l \le T_k \le l$.) Once W_k has been found, T_k can be computed via a brute force discrete log search. As a small optimization, one can arrange up front to have the T_k satisfy a known linear equation, in which case the server computer can compute one of them directly from the others without a discrete log search.

Summary. A summary of the steps under the discrete log protocol method performed by the system 100 is:

1. The authorities jointly generate the system parameters p, g, G_k, y_k, n_k, and h (for $1 \le k \le N$).

2. Voter V_i posts a ballot to the ballot box

$$\{(x_{i,k}y_{i,k})\}_{k=1}^{N} = \{(g^{\alpha(i,k)}, h^{\alpha(i,k)}G_k^{b(i,k)})\}_{k=1}^{N}$$

where $\alpha(i,k) \in_R Z_p, b(i,k) \in \{y_k(yes), n_k(no)\}$ and $1 \le k \le N$.

3. Voter V_i accompanies the ballot with a non-interactive proof of validity, POV_i, to ensure that the ballot is well-formed. POV_i is checked by the authorities. This check can be completed either during polling or after the polls close.

4. When the polls close, the encrypted tally, $(X, Y)_k = \left(\prod_{i=1}^{l} x_{i,k}, \prod_{i=1}^{l} y_{i,h}\right)$, is formed.

5. The authorities jointly execute the decryption protocol described above for $(X,Y)_k$ to obtain $W_k = G_k^{T_k} = Y_k / X_k^s$, where T_k satisfies equation 12, $1 \le k \le N$.

6. Find T_k by computing $G_k^T, T = py_k + (l-p)n_k, 0 \le p \le l$, until $W_k = G_k^T$; $1 \le k \le N-1$.

7. Compute T_N, either by the previous search method, or by solving a known linear equation involving T_k, $1 \le k \le N-1$.

8. For the unique value pk satisfying $T_k = p_k y_k + (1 - p_k) n_k$, publish the values $N_k(y) = p_k$ and $N_k(n) = 1 - p_k$ as the total 'yes votes' and 'no votes' respectively for ballot option k.

Universal Verification. To verify the election in the discrete log context, the following protocol is executed:

1. Using tally, T_k, compute $\overline{W_k} = G_k^{T_k}$ and check that $\overline{W_k} = W_k$.

2. For voter V_i, check POV_i. This proves that each ballot is well-formed.

3. Compute the product $\overline{(X,Y)_k} = \left(\prod_{i=1}^{I} x_{i,k}, \prod_{i=1}^{I} y_{i,k}\right)$ and check that $\overline{(X,Y)_k}$ is equal to the published values $(X,Y)_k$

4. Check proofs of validity, submitted by the authorities, for the decryption protocol described above. This proves that $W_k = y_k / X_k^s$ without reconstructing the authorities' private key s.

5. Verify that $N_k(y)$ and $N_k(n)$ satisfy equation 12.

Furthermore, voter V_i can optionally be given the random number used to encrypt option k (*i.e.*, $\alpha_{i,k}$). The random number can later be used to regenerate the ciphertext $\left(x_{i,k}, y_{i,k}\right) = \left(g^{\alpha_{i,k}}, h^{\alpha_{i,k}} G_k^{b_k}\right)$ and compare it to the contents of the ballot box. This has the disadvantage that $\alpha_{i,k}$ is a receipt for option k and could be used to convince a coercer.

4.2 Elliptic Curve Formulation

We now describe the election scheme in an elliptic curve context for a publicly known finite field Z_p, elliptic curve E defined over Z_p and a base point $g \in E$. The security of the scheme depends on the security of computing discrete logs on elliptic curves which is likely to be more intractable than computing discrete logs in finite fields. See *e.g.*, N. Koblitz, A Course in Number Theory and Cryptography, 2nd edition, Springer, 1994; A. M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, Advances in Cryptology - EUROCRYPT '84, Lecture Notes in Computer Science, Springer-Verlag, 1984.

Election preparation. The system parameters E and points g, $G_k \in E$, as well as y_k, $n_k \in Z_{q_k}$ (where $1 \le k \le N$ ballot options, and q_k is the order of the cyclic sub-

group generated by G_k) are generated jointly by a subset of the authorities. As in the Z_p setting, one can choose to simplify the implementation by choosing the N generators, G_k, as well as the N pairs y_k, n_k; so that they are independent of k. More specifically, it is most convenient to always use the values $y_k = y_0 = 1$ and $n_k = n_0 = -1$, or $y_k = y_0 = 1$ and $n_k = n_0 = 0$. The parameter $h \in \langle g \rangle$ is also generated using the secret sharing protocol in essentially the same way it was generated in the Z_p setting.

Casting a ballot. To cast a ballot, an encryption is made of each ballot option. The message is $b_k G_k$, where $b_k \in \{y_k(yes), n_k(no)\}$ for option k. An encrypted ballot is then of the form

$$B_i = \{(x_{i,k}, y_{i,k})\}^N_{k=1} = \{(\alpha(i,k)\, g, \alpha(i,k)\, h + b(i,k)\, G_k)\}^N_{k=1} \quad 1 \le k \le N. \tag{13}$$

where $\alpha(i,k) \in \langle g \rangle^*$ is chosen randomly by voter V_i, and kept secret. In addition to the ballot, a non-interactive proof of validity is submitted according to the protocol described above.

Tallying. To tally the election, the votes are combined, exploiting the homomorphic property described above. For elliptic curves, this involves summing all votes to obtain a single encrypted tally for option k, $(X,Y)_k$.

The authorities jointly execute the decryption protocol described above for $(X,Y)_k$ to obtain $W_k = T_k G_k = Y_k - sX_k$, where T_k again satisfies equation 12. Again, once W_k has been found, T_k can be computed via a brute force search. The optimization of the Z_p setting also applies here.

Summary. A summary of the elliptic curve protocol method is:

1. The authorities jointly generate the system parameters E, g, G_k, y_k, n_k, and h (for $1 \le k \le N$).

2. Voter V_i posts a ballot to the ballot box

$$\{(x_{i,k}, y_{i,k})\}^N_{k=1} = \{(\alpha(i,k)\, g, \alpha(i,k)\, h + b(i,k)\, G_k)\}^N_{k=1}$$

where $\alpha_{i,k} \in_R Z_p$, $b(i,k) \in \{y_k(yes), n_k(no)\}$, and $1 \le k \le N$.

3. Voter V_i accompanies the ballot with a non-interactive proof of validity, POV_i, to ensure that the ballot is well-formed. POV_i is checked by the authorities. This check can be completed either during polling or after the polls close.

4. When the polls close, the encrypted tally, $(X,Y)_k = (\sum_{i=1}^{I} x_{i,k}, \sum_{i=1}^{I} y_{i,k})$, is formed.

5. The authorities jointly execute the decryption protocol described above for $(X,Y)_k$ to obtain $W_k = T_k G_k = Y_k - sX_k$, where T_k satisfies equation 12.

6. Find T_k by computing $T G_k$, $T = p\, y_k + (l - p)\, n_k$ $0 \le p \le l$, until $W_k = T G_k$, $1 \le k \le N - 1$.

7. Compute T_N, either by the previous search method, or by solving a known linear equation involving T_k, $1 \le k \le N - 1$.

8. For the unique value p_k satisfying $T_k = p_k\, y_k + (l - p_k)\, n_k$, publish the values $N_k(y) = p_k$ and $N_k(n) = l - p_k$ as the total 'yes votes' and 'no votes' respectively for ballot option k.

Universal verification. To verify the election in the elliptic curve context, the following protocol is executed:

1. Using tally, T_k, compute $\overline{W_k} = T_k G_k$ and check that $\overline{W_k} = W_k$.

2. For voter V_i, check POV_i. This proves that each ballot is well-formed.

3. Compute the sum $(X,Y)_k = \left(\sum_{i=1}^{I} x_{i,k}, \sum_{i=1}^{I} y_{i,k}\right)$ and check that $\overline{(X,Y)_k} = (X,Y)_k$.

4. Check proofs of validity, submitted by the authorities, for the decryption protocol of Section 2.5. This proves that $W_k = T_k G_k$ without reconstructing the authorities' private key s.

5. Verify that $N_k(y)$ and $N_k(n)$ satisfy equation 12.

Furthermore, voter V_i can optionally be given the random number used to encrypt option k (*i.e.*, $\alpha(i,k)$). The random number can later be used to regenerate the ciphertext $(x_{i,k}, y_{i,k}) = (\alpha(i,k)g,\ \alpha(i,k)h + b(i,k)G_k)$ and compare it to the contents of the ballot box. This has the disadvantage that $\alpha(i,k)$ is a receipt for option k and could be used to convince a coercer.

4.3 General Group Formulation

Both the discrete log (Z_p) formulation and the Elliptic Curve formulation are special cases of the following general setting.

Let ε be an encoding of an abstract group G into some finite alphabet Σ. So

$$\varepsilon : G \to \Sigma^*$$

Choosing any fixed element $g \in \backslash G$ induces an encoding $\varepsilon_g : Z \to \Sigma^*$ via the rule

$$\varepsilon_g(n) = \varepsilon(g^n)$$

in fact, we have $\varepsilon_g : Z_q \to \Sigma^*$, where $q = |\langle g \rangle|$.

In the previous subsections, $G = Z_p$, E respectively. In both cases, the encoding, ε of G was not explicitly stated, but it was implied. For example, little thought is necessary to realize that each voter, V_i, actually submits a bit-encoding of his ballot; B_i (see equations 11 and 13). (On digital computers now in use, $\Sigma = \{0, 1\}$.)

Having made a note of this simple, but important point, it can now be seen that both the Z_p setting and the Elliptic Curve setting of the previous subsections can be interpreted in the general framework above. In the case of Z_p, the binary group operation was written in *multiplicative* notation, while in the Elliptic Curve case, it was written in *additive* notation. Aside from this notational difference, the description of the multi-way election scheme in both settings is effectively identical. In each case, only the abstract group properties, and the particular bit-representation of the group were used. Thus, *implementation of the multi-way election scheme is not limited to the two settings previously discussed.*

Referring to Figure 4, a general, overall, routine 400 for performing a multi-way election is shown. In step 402, an organization determines initial cryptographic parameters, such as a group (*e.g.*, Z_p or elliptic curve), a subgroup generator g, a size of the group G and a size of the generated subgroup q. This information is provided to a number n of tabulation authorities, whereby a subset of these authorities K are needed to decrypt ballots, as explained below.

In step 404, the tabulation authorities T produce a public key h. In step 406, a vote collection center, such as an organization or group operating the server computer system 108, employs the cryptographic parameters to encode a ballot and digitally sign that ballot. When a voter requests a ballot and provides proper authentication of any necessary registration information, the vote collection center provides an encoded ballot to the voter. More information on registering voters may be found in U.S. Patent No. _____, filed March 24, 2000, entitled "Method, Article and

Apparatus for Registering Registrants, Such as Voter Registrants," and assigned to the assignee of this application. Under step 406, the vote collection center also digitally signs or hashes each ballot distributed to a voter.

In step 408, each voter verifies the digital signature or hash function result provided with the ballot to ensure the ballot has not been altered and was received from the appropriate vote collection center. Under step 408, each voter makes a decision d for each of the ballot options i, or each ballot option has one of two choices (yes or no, or G^y or G^n). The voter computer 102 then produces encrypted decisions d_i, and provides a validity proof based on the encrypted decisions. The validity proof ensures that some number k of options were correctly voted on so that all constraints on the ballot are satisfied. The proof of the validity may include the polynomial result s_i, or the coefficients of the polynomial $T(x)$ (as explained below). The voter also includes his or her digital signature with the ballot and submits it to the vote collection center.

In step 410, the vote collection center verifies the voter's eligibility and digital signature and verifies the proof of validity included with the ballot. If both of these checks pass, then the ballot is added to a "ballot box" which may be both a database containing all these ballots, as well as ballots written permanently to a write-once, read-many storage device, which is described in more detail in U.S. Patent No. _____, filed March 24, 2000, entitled "Electronic Voting Scheme Employing Permanent Ballot Storage" and assigned to the assignee in this application. After receiving a ballot (and possibly after verifying it), the vote collection center under step 410 provides a digitally signed receipt to the voter. Under an alternative embodiment, the vote collection center provides the digitally signed receipt to one or more of the tabulation authorities (as shown by the broken line 409 in Figure 4). The tabulation authorities may then add their own digital signatures before forwarding the receipt to the voter. Thus, the tabulation authorities may retain a collection of receipts provided by the vote collection center to verify that all ballots provided by voters have been included in the final tally.

In step 412, the vote collection center aggregates the encrypted decisions after the polls close and produces an encrypted tally A_i, a copy of which is sent to each of the tabulation authorities T. In step 414, each of the tabulation authorities produce their decryption share B_i (k), to the vote collection center. The tabulation authorities do not

transmit their secret keys. In steps 416 and 418, the vote collection center receives the decryption shares from the tabulation authorities and decrypts a result S_i.

In step 420, the vote collection center either by a brute force method determines the log_G of the values S_i, or does a search of a previously computed and stored table of such computations, as described below. In step 422, the vote collection center publishes the encrypted voter ballots and signatures, voter validity proofs, authority validity proofs and results of the tally or election. Each voter may then verify that his or her ballot was included in the tally. In step 424, a third party or external audit/review organization may verify the results to ensure the election was performed without fraud.

Under one embodiment, after each voter submits his or her encrypted ballot under step 408 to the vote collection center, the vote collection center under step 410 may issue a receipt to the voter that includes the digital signature of the vote collection center. This receipt, with a digital signature, provides indication or assurance to the voter that his or her ballot has been received. Under an alternative embodiment, the receipt provided by the vote collection center first passes through the tabulation authorities (*i.e.*, the authority computers 114), who each add their own digital signatures to the receipt. Thereafter, the voter receives a receipt having digital signatures from not only the vote collection center but also the tabulation authorities. This allows the tabulation authorities to check receipts received from the vote collection center to ensure that the vote collection center has not deleted or omitted ballots from the resulting tally.

4.4 Tabulation Search Enhancement

One of the more time consuming operations required in the tabulation phase of the multi-way election scheme is the search for each T_k. To make matters worse, this piece of the tabulation can not be started until after the polls are closed, unlike the validity verification and aggregation pieces, which can be carried out on each vote as it is received. This can cause a significant delay in posting the election results.

This situation can be improved by making a time/space tradeoff. In other words, to improve processing time, memory or data storage space is required as explained below. Assume it is known sometime prior to tabulation, an upper bound L on the total number of votes cast in any precinct. For any $0 \leq \alpha \leq 1$, the server computer can

precompute and store L^{α} quantities that allow it to compute each T_k (during the tabulation search stage) using at most $L^{\alpha} + L^{(1-\alpha)}$ multiplications and at most $\alpha \log (L)\ L^{(1-\alpha)}$ comparisons, instead of the L multiplications and L comparisons that are required by the straightforward approach. In other words, by precomputing such values, and storing them in an indexed fashion into a look-up table in the database 110, the server computer system 108 may more readily compute a tally under step 420 above. The procedure is described as follows:

INPUT: A generator, g of a cyclic group, G, an element $h \in G$ and an upper bound L on $\log_g h$. (That is, knowledge that there exists an s with $0 \leq s \leq L$ such that $g^s = h$.)

OUTPUT: The value of s above.

PROCEDURE:

Precomputation. (Done *once* before tabulation.)

1. Set $m = \lceil L^{\alpha} \rceil$ or the smallest integer greater than or equal to L^{α}.
2. Construct and store a sorted list of elements (j, g^j) for $0 \leq j \leq m$, sorted on the value of the second index.
3. Compute and store $\delta = g^{-m}$.

Search. (Done at tabulation time to find each T_k.)

Set $\gamma = h$, $\mu = \lceil L(1 - \alpha) \rceil$, and $f = 0$. For $0 \leq i \leq \mu - 1$ execute the following until $f = 1$.

If (j,γ) is in the precomputed table, set $s = im + j$, and $f = 1$ (return).

Otherwise, set $\gamma = \delta\gamma$ (iterate loop).

Note that this tabulation search enhancement applies to the general group setting, since no special properties of Z_p or Elliptic Curves were used. Even greater efficiencies are obtained for the typical election parameter choices where G_k, y_k, and n_k are all chosen to be independent of k. In this case, the server computer need only precompute *one* table regardless of the number of ballot choices.

Referring to Figure 5, a routine 500 performed by the server computer system 108 illustrates the above tabulation search enhancement. Under step 502, the server computer system 108 constructs and stores a sorted list of pairs (j, g^j) before all electronic ballots are received from voters or before the decrypted aggregated tally is received. The sorted list is sorted based on the second coordinate (*i.e.*, g^j). The number

of values chosen are set at some memory storage space constraint. Thus, under step 502, the server computer 108 precomputes values g^j for values of j from 1 to a number chosen to match the constraint.

Under step 504, the server computer system 108 successively generates values of the form $h\, g^{-im}$ after the decrypted tally is received. In step 506, the server computer system 108 determines if the generated value from step 504 matches a previously computed and stored value in the list (based on the second coordinate). If not, then the routine 500 loops back to step 504. If yes, then the decrypted tally results under step 508.

The following is a detailed example of the homomorphic voting protocol. In order to maximize the clarity of our presentation, several of the basic parameters used - for example, the number of voters, and the size of the cryptographic parameters - are much smaller than would be the case in practice. Nevertheless, the example helps explain important aspects of the system. The operations that must be performed in a full scale election occur in this downsized example, and are documented here. (Since the tabulation speedup methodology can only be illustrated on large quantities of data, it will not be presented here. Instead, a simple search is used in the final stage of tabulation.)

5. Example: Election Initialization

An organization must first agree on a ballot (a set of questions and allowable answers, a page layout, etc.), a group of eligible voters, a standard information encoding scheme (a published ASN1 standard, HEX encoded, for example), and the basic cryptographic numerical parameters. It also agrees on who will be the set of tabulation authorities, and the threshold, t, for decryption. Optionally, it may also agree on a "ballot signing authority", whose responsibility it is to authenticate, via digital signature, the final encoded form of the ballot. Once agreement has been reached, all of this information is encoded in the agreed upon format, and this data string is digitally signed by whatever parties are required to represent, as a group, the consensus of the organization.

Under the following example, two voters vote on colors for a new flag, where a maximum number of two colors per voter may be selected. Three tabulation

authorities are used to decrypt the ballot, where only two of the three authorities are necessary for decryption. For the purposes of this example, we will assume that the essential parameters have been chosen as follows:

Ballot

- **One Question**
  - **Question 1 Text:**
  - **Number of answers/choices:** 4
    - **Answer 1 Text:** *Blue*
    - **Answer 2 Text:** *Green*
    - **Answer 3 Text:** *Red*
    - **Answer 4 Text:** *Black*
  - **Max Positive:** (Maximum "yes" answers/votes allowed) 2

Eligible Voters: 2

1. **Name:** $V1$
2. **Name:** $V2$

Tabulation Authorities: 3

1. **Name:** $A1$, **Index:** 1
2. **Name:** $A2$, **Index:** 2
3. **Name:** $A3$, **Index:** 3

Tabulation Threshold: $t = 2$

Cryptographic Parameters

- **Group Arithmetic:** Integer mutiplicative modular arithmetic
- **Prime Modulus:** $p = 47$
- **Subgroup Modulus:** $q = 23$
- **Generators:** $g = 2$ and $G = 3$. (From this it can be easily computed the inverse or "no/against" vote for G, *i.e.*, $G^{-1} = 16$.)
- **Public Key:** $h = g^s$ where s is unknown. h is created by having the tabulation authorities execute the *threshold secret sharing scheme* explained herein. For sake of this example, let us say that h is determined to be 7. (Remember that this is a "toy" example. In this case $s = 12$ can be determined by brute force, since the cryptographic parameters were chosen to be too small

to be secure. In a real example, this would not be the case; it would be computationally impossible to determine s from the known value of h.)

A separate *signed, encoded ballot* is also formed. The signature on this ballot is either the same group signature used on the election parameters, or the signature of the appointed "ballot signing authority", but the data contained in it may be a subset of the above data. At a minimum, the ballot encodes the values of the numerical cryptographic parameters p, q, g, the exact sequence of questions, question text, allowable question answers, and question "max positives". Once this encoded ballot is signed, the election is ready to take place.

6. Example: Voting

6.1 Ballot Request and Transmission

Sometime during the preset official polling time, the voters, V_1 and V_2 each independently submit a *ballot request,* along with authentication information. Assuming that authentication information passes the required tests, a ballot is delivered to the respective requester. Further information on authentication schemes may be found in U.S. Patent Application No. _____, filed March 24, 2000, entitled "Method, Article and Apparatus for Registering Registrants, Such as Voter Registrants," and assigned to the same assignee of this invention. On receiving the ballot, each voter first checks the signature to be sure it is valid. Assuming the signature check passes, the voter then decodes the ballot. From the ballot, the voters are able to determine that the expected response is the standard encoding of a particular sequence of *seven* distinct data elements. These are (in their precise order):

1. A pair of integers (l_1, r_1) with $0 \le l_1, r_1 < 47$ indicating (in encrypted form) the voter's choice for the answer *"Blue"*. This is an encryption of one of two values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked".

2. A pair of integers (l_2, r_2) with $0 \le l_2, r_2 < 47$ indicating (in encrypted form) the voter's choice for the answer *"Green"*. This is an encryption of one of two

values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked".

3. A pair of integers (l_3, r_3) with $0 \le l_3, r_3 < 47$ indicating (in encrypted form) the voter's choice for the answer *"Red"*. This is an encryption of one of two values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked".

4. A pair of integers (l_4, r_4) with $0 \le l_4, r_4 < 47$ indicating (in encrypted form) the voters choice for the answer *"Black"*. This is an encryption of one of two values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked".

5. A pair of integers (l_5, r_5) with $0 \le l_5, r_5 < 47$ indicating (in encrypted form) the voter's choice for the answer *"Do Not Care 1"*. This is an encryption of one of two values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked". This value will never affect the final tally of any of the four "real" responses - "Blue", "Green", "Red" or "Black". It is only present to support the proof of validity that the voter must also submit.

6. A pair of integers (l_6, r_6) with $0 \le l_6, r_6 < 47$ indicating (in encrypted form) the voter's choice for the answer *"Do Not Care 2"*. This is an encryption of one of two values, which can be interpreted as "for or against" or "positive or negative" or "checked or unchecked". This value will never affect the final tally of any of the four "real" responses - "Blue", "Green", "Red" or "Black". It is only present to support the proof of validity that the voter must also submit.

7. A *proof of validity* which demonstrates that among the six encrypted values above, there are *exactly two* which are encryptions of the value "checked" (also known as "for" or "positive"). This proof *reveals no information about which two of the six are as such.* In the case of this election, the form of the proof is a sequence of 42 numbers, all in the range (0, 47) and in precise order. In other elections, a validity proof may consist of more or fewer numbers in total, but the count will be constant for each submitted ballot in a fixed election, and the numbers will always be submitted in precise order.

6.2 Decision Encryption

For the sake of this example, let us assume that V_1 wishes to cast a vote for "Blue" and a vote for "Black", and that V_2 wishes to cast a vote for "Black" only.

6.2.1 V_1 Encrypted Decisions

1st Decision

V_1 generates an exponent value e_1, which is selected at random from the subgroup modulus of 23 integers, *i.e.*, $e_1 \in_R Z_{23}$. For sake of example, say $e_1 = 5$. Since "Blue" is the first answer possibility, and since V_1 wishes to vote "for" the choice "Blue", the first encrypted decision is given by

$$(x_1, y_1) = (2^5, 7^5 \times 3) = (32, 37) \quad (1)$$

2nd Decision

V_1 generates $e_2 \in_R Z_{23}$. For sake of example, say $e_2 = 13$. Since "Green" is the second answer possibility, and since V_1 wishes to vote "against" the choice "Green", the second encrypted decision is given by

$$(x_2, y_2) = (2^{13}, 7^{13} \times 16) = (14, 24) \quad (2)$$

3rd Decision

V_1 generates $e_3 \in_R Z_{23}$. For sake of example, say $e_3 = 11$. Since "Red" is the third answer possibility, and since V_1 wishes to vote "against" the choice "Red", the third encrypted decision is given by

$$(x_3, y_3) = (2^{11}, 7^{11} \times 16) = (27, 12) \quad (3)$$

4th Decision

V_1 generates $e_4 \in_R Z_{23}$. For sake of example, say $e_4 = 20$. Since "Black" is the fourth answer possibility, and since V_1 wishes to vote "for" the choice "Black", the fourth encrypted decision is given by

$$(x_4, y_4) = (2^{20}, 7^{20} \times 3) = (6, 17) \quad (4)$$

5th Decision

This is an answer of type "Do Not Care". V_1 generates $e_5 \in_R Z_{23}$. For sake of example, say $e_5 = 17$. Since V_1 has voted "for" two answer possibilities, and since 2 is

the "Max Positive" for this question, V_1 must encrypt this as a choice "against". So the fifth encrypted decision is given by

$$(x_5, y_5) = (2^{17}, 7^{17} \times 16) = (36, 2) \quad (5)$$

6^{th} Decision

This is another answer of type "Do Not Care". V_1 generates $e_6 \in_R Z_{23}$. For sake of example, say $e_6 = 10$. Since V_1 has voted "for" two answer possibilities, and since 2 is the "Max Positive" for this question, V_1 must encrypt this as a choice "against". So the sixth encrypted decision is given by

$$(x_6, y_6) = (2^{10}, 7^{10} \times 16) = (37, 42) \quad (6)$$

6.2.2 V_2 Encrypted Decisions

1^{st} Decision

V_2 generates $e_1 \in_R Z_{23}$. For sake of example, say $e_1 = 18$. Since "Blue" is the first answer possibility, and since V_2 wishes to vote "against" the choice "Blue", the first encrypted decision is given by

$$(x_1, y_1) = (2^{18}, 7^{18} \times 16) = (25, 14) \quad (7)$$

2^{nd} Decision

V_2 generates $e_2 \in_R Z_{23}$. For sake of example, say $e_2 = 14$. Since "Green" is the second answer possibility, and since V_2 wishes to vote "against" the choice "Green", the second encrypted decision is given by

$$(x_2, y_2) = (2^{14}, 7^{14} \times 16) = (28, 27) \quad (8)$$

3^{rd} Decision

V_2 generates $e_3 \in_R Z_{23}$. For sake of example, say $e_3 = 21$. Since "Red" is the third answer possibility, and since V_2 wishes to vote "against" the choice "Red", the third encrypted decision is given by

$$(x_3, y_3) = (2^{21}, 7^{21} \times 16) = (12, 8) \quad (9)$$

4^{th} Decision

V_2 generates $e_4 \in_R Z_{23}$. For sake of example, say $e_4 = 16$. Since "Black" is the fourth answer possibility, and since V_2 wishes to vote "for" the choice "Black", the fourth encrypted decision is given by

$$(x_4, y_4) = (2^{16}, 7^{16} \times 3) = (18, 16) \quad (10)$$

5th Decision

This is an answer of type "Do Not Care". V_2 generates $e_5 \in_R Z_{23}$. For sake of example, say $e_5 = 4$. Since V_2 has voted "for" only one possibility, and since 2 is the "Max Positive" for this question, V_2 must encrypt one of the "Do Not Care" answer choices as "for" and one "against". We may as well assume that he chooses to encode this one as "for", and the next one as "against". (Implementation of the alternative is only trivially different.) So the fifth encrypted decision is given by

$$(x_5, y_5) = (2^4, 7^4 \times 3) = (16, 12) \quad (11)$$

6th Decision

This is another answer of type "Do Not Care". V_2 generates $e_6 \in_R Z_{23}$. For sake of example, say $e_6 = 9$. As noted above in the description of the previous decision, we may assume that this answer choice needs to be "against". So the sixth encrypted decision is given by

$$(x_6, y_6) = (2^9, 7^9 \times 16) = (42, 6) \quad (12)$$

6.3 Proof of Validity

6.3.1 V_1 Proof of Validity

Voter V_1 must prove that two of his encrypted decisions are encryptions of "for" and that four of his encrypted decisions are encryptions of "against". (Recall that nothing about the form of an encrypted decision prevents V_1 from encrypting a value other than these two special values, so the second half of this criteria *is not* implied by the first. The full criteria is, however, implied if V_1 can show that *at least* two of his encrypted decisions are encryptions of "for" and *at least* four of his encrypted decisions are encryptions of "against". Thus it is natural that the proof consist of two parts.

Proof Part 1: At least two of the encrypted decisions are encryptions of "for".

1. V_1 sets $\alpha_1, \ldots, \alpha_6$ to be the exponents he used to encrypt his corresponding decision. So in this example:

$$\alpha_1 = 5, \alpha_2 = 13, \alpha_3 = 11, \alpha_4 = 20, \alpha_5 = 17, \alpha_6 = 10 \quad (13)$$

2. V_1 generates $\omega_1, \omega_2 \in_R Z_{23}$, $r_2, r_3, r_5, r_6 \in_R Z_{23}$, $s_2, s_3, s_5, s_6 \in_R Z_{23}$ randomly. For this example we take

$$\omega_1 = 4, \omega_4 = 13 \quad (14)$$

$$r_2 = 16, r_3 = 17, r_5 = 21, r_6 = 9$$

$$s_2 = 12, s_3 = 4, s_5 = 15, s_6 = 8$$

3. V_1 computes corresponding values

$$\begin{aligned}
a_1 = g^{\omega}{}_1 &= 2^4 = 16 \\
a_2 = g^{r}{}_2 x_2^{-s}{}_2 &= 2^{16} \times 14^{11} = 12 \\
a_3 = g^{r}{}_3 x_3^{-s}{}_3 &= 2^{17} \times 27^{19} = 3 \\
a_4 = g^{\omega}{}_4 &= 2^{13} = 14 \\
a_5 = g^{r}{}_5 x_5^{-s}{}_5 &= 2^{21} \times 36^{8} = 3 \\
a_6 = g^{r}{}_6 x_6^{-s}{}_6 &= 2^{9} \times 37^{15} = 12 \\
b_1 = h^{\omega}{}_1 &= 7^4 = 4 \\
b_2 = h^{r}{}_2 (y_2/G)^{-s}{}_2 &= 7^{16} \times (24/3)^{11} = 25 \\
b_3 = h^{r}{}_3 (y_3/G)^{-s}{}_3 &= 7^{17} \times (12/3)^{19} = 7 \\
b_4 = h^{\omega}{}_4 &= 7^{13} = 25 \\
b_5 = h^{r}{}_5 (y_5/G)^{-s}{}_5 &= 7^{21} \times (2/3)^{8} = 18 \\
b_6 = h^{r}{}_6 (y_6/G)^{-s}{}_6 &= 7^{9} \times (42/3)^{15} = 16
\end{aligned}$$

4. V_1 uses an agreed upon hash function H to compute $c \in_R Z_{23}$

$$c = H(\{x_i, y_i, a_i, b_i\}) \; 1 \le i \le 6 \quad (15)$$

Since many choices of the hash function are possible, for this example we can just pick a random value, say

$$c = 19 \quad (16)$$

5. V_1 computes the interpolating polynomial $P(x)$ of degree 6 - 2 = 4. The defining properties of P are

$$\begin{aligned}
P(0) &= c = 19 \quad (17) \\
P(2) &= s_2 = 12 \\
P(3) &= s_3 = 4 \\
P(5) &= s_5 = 15
\end{aligned}$$

$$P(6) = s_6 = 8$$

The technique for computing $P(x) = \Sigma_{j=0}^{4} z_j x^j$ is standard polynomial interpolation theory. Hence, rather than go through the steps of the computation here, we simply state the result:

$$P(x) = 42x^4 + 23x^3 + 2x^2 + 11x + 19 \tag{18}$$

$$z_0 = 19 \quad z_1 = 11 \quad z_2 = 2 \tag{19}$$
$$z_3 = 23 \quad z_4 = 42$$

6. V_1 computes the values

$$s_1 = P(1) = 3 \qquad s_4 = 5 \tag{20}$$

$$r_1 = \omega_1 + \alpha_1 s_1 = 4 + 5s_1 = 19$$

$$r_4 = \omega_4 + \alpha_4 s_4 = 13 + 20s_4 = 21$$

7. Part 1 of V_1's validity proof consists of the 18 numbers

$$\{a_k, b_k, r_k\}_{k=1}^{6} \tag{21}$$

and the four numbers

$$\{z_k\}_{k=1}^{4} \tag{22}$$

in precise sequence. Notice that z_0 need not be submitted, since it is computable from the other data elements submitted using the publish hash function H. The value $P(0)$ may be readily computed based on the result of the hash function transmitted to the verifying party. Thus, this example explains another embodiment over that described above in section 1, which results in a 8% to 10% reduction in the size of each electronic ballot. In other words, the results of the polynomials s_j are not transmitted (s_1 and s_4, in this example, for a savings of four numbers for the two parts of the proof). Instead, this embodiment transmits only the coefficients for the polynomials $P(x)$'s. Rather than forcing the receiver party or voting authority to again perform standard polynomial interpolations as the voting party had to do previously, such redundant processing is avoided in this embodiment.

Proof Part 2: At least four of the encrypted decisions are encryptions of "against".

The proof of part 2 can be obtained by following the same steps as were given in part 1 with the following simple substitutions:

1. The indices 1 and 4 are now treated like the indices 2, 3, 5 and 6 were treated in the Proof of Part 1 above, and visa versa.

2. The value G (3) is interchanged with the value G^{-1} (*i.e.*, 16) and visa versa.

3. The degree of the interpolating polynomial P is now 6 - 4 = 2 instead of 4.

4. The size of the proof is now 20 = 18 + 2 numbers, rather than 22 = 18 + 4 numbers, as a result in the change of degree of the interpolation polynomial (not transmitting z_3 and z_4, in this example).

6.3.2 V_2 Proof of Validity

The construction of V_2's proof of validity follows essentially the same steps as were presented for V_1's proof of validity, so we only highlight the significant differences:

1. For V_2 the indices 4 and 5 play the role that the indices 1 and 4 played for V_1.

2. For V_2 the indices 1, 2, 3 and 6 play the role that the indices 2, 3, 5 and 6 played for V_1.

3. The randomly generated quantities are obviously different, in fact independent.

6.4 Vote Submission

Having computed the required six numerical pairs in sequence, and the corresponding proof of validity (parts 1 and 2), V_1 and V_2 each encode these elements, in sequence, as defined by the standard encoding format. The resulting sequences (belonging to V_1 and V_2 respectively) form each voter's encoded encrypted decision sequence, or *voted ballot*. Each voter then digitally signs his voted ballot with his private

signing key. For each voter, V_i, (in this example $i = 1, 2$) the resulting combination of V_i's voted ballot, and his digital signature (more precisely, the standard encoding of these two elements) forms his **signed voted ballot**. Finally, each voter transmits his signed voted ballot back to the data center collecting the votes.

7. Example: Tabulation

7.1 Verification

Before the voted ballots are included in the collection to be tallied, they must first pass two verification checks.

Signature Check

First, the digital signature is checked to be sure it matches that of an eligible voter; in fact, it must match the signature of an eligible voter *who has not already voted* (i.e. has not already submitted a vote).

Validity Verification

The voted ballot must pass verification of the included proof of validity.

Digital signature checking is a standard technique. For validity verification, the voted ballot must pass two checks, corresponding to parts 1 and 2 of the proof of validity. For part 1, the steps consist of:

1. Compute the zero order coefficient of P, z_0, by

$$z_0 = H(\{x_i, y_i, a_i, b_i\})_{i=1}^{6} = 19 \tag{23}$$

2. For each $1 \le j \le 6$ both sides of the equations

$$\begin{aligned} a_j &= g^{r_j} x_j^{-P(j)} \\ b_j &= h^{r_j} (y_j/G)^{-P(j)} \end{aligned} \tag{24}$$

are evaluated. If equality fails in *any* of these, verification fails. This ballot is not included in the ballot box.

In the case of this example, equation 23 holds since this was a basic assumption made about the arbitrary hash function H. (See equations 15 and 16.) As for the set of equations labeled 24, in this example they reduce to checking:

$$16 = 2^{19} \times 32^{-3} \qquad 4 = 7^{19} \times (37/3)^{-3}$$

$$\begin{aligned} 12 &= 2^{16} \times 14^{-12} & 25 &= 7^{16} \times (24/3)^{-12} \\ 3 &= 2^{17} \times 27^{-4} & 7 &= 7^{17} \times (12/3)^{-4} \\ 14 &= 2^{21} \times 6^{-5} & 25 &= 7^{21} \times (17/3)^{-5} \\ 3 &= 2^{21} \times 36^{-15} & 18 &= 7^{21} \times (2/3)^{-15} \\ 12 &= 2^{9} \times 37^{-8} & 16 &= 7^{9} \times (42/3)^{-8} \end{aligned} \qquad (25)$$

Verification of each of these 12 equations is a tedious but trivial matter. We leave it up to the reader to check them to his satisfaction.

The steps required to verify part 2 are the same except that $G^{-1} = 16$ is used in place of $G = 3$ and visa versa.

7.2 Vote Aggregation

The collection of votes that pass validity verification constitute the ballot box. For this example, we will assume that both the ballot belonging to V_1 and the ballot belonging to V_2 passed. Thus, in this example, the ballot box will consist of four pairs of encrypted decisions - each of these corresponding, in sequence, to a answer choice or option *in the same sequence.*

Ballot Box

BLUE (32, 37) ; (25, 14)

GREEN (14, 24) ; (28, 27)

RED (27, 12) ; (12, 8)

BLACK (6, 17) ; (18, 16)

An encrypted tally is formed for each of these by multiplying the pairs together component wise:

Encrypted Tallies

BLUE $(U_1, T_1) = (1, 1) = (32 \times 25, 37 \times 14)$

GREEN $(U_2, T_2) = (16, 37) = (14 \times 28, 24 \times 27)$

RED $(U_3, T_3) = (42, 2) = (27 \times 12, 12 \times 8)$

BLACK $(U_4, T_4) = (14, 37) = (6 \times 18, 17 \times 16)$

(Recall: operations are performed, in this example (mod 47).

7.3 Decryption

The threshold secret scheme is now executed by the authorities. The end result of this is to compute the values U_j^s for $1 \le j \le 4$, where s is the secret value, *without revealing information about s to anyone*. Since this technique is standard, we do not illustrate it here, rather we note what the values must be (since in this toy example we do know $s = 12$ - see above).

$$\begin{aligned} U_1^s &= 1^{12} = 1 \\ U_2^s &= 16^{12} = 4 \\ U_3^s &= 42^{12} = 18 \\ U_4^s &= 14^{12} = 25 \end{aligned} \tag{26}$$

From this we compute

$$\begin{aligned} V_1 = T_1/U_1^s &= 1/1 = 1 \\ V_2 = T_2/U_2^s &= 37/4 = 21 \\ V_3 = T_3/U_3^s &= 2/18 = 21 \\ V_4 = T_4/U_4^s &= 37/25 = 9 \end{aligned} \tag{27}$$

7.4 Exponent Search

By simple search (we would use the *Tabulation Search Enhancement* for large elections) we obtain (recall $G = 3$ for this election)

$$\begin{aligned} V_1 &= G^0 \\ V_2 &= G^{-2} \\ V_3 &= G^{-2} \\ V_4 &= G^2 \end{aligned} \tag{28}$$

7.5 Linear Equation Solution

Each exponent in equation 25 is exactly the difference between the number of "for" votes cast and the number of "against" votes cast for the corresponding answer

choice. Since we know the sum of the number of "for" votes and the number of "against" votes is always 2 (the number of ballots submitted), we can compute

BLUE 1 vote "for" ; 1 vote "against"

GREEN 0 votes "for" ; 2 votes "against"

RED 0 votes "for" ; 2 votes "against"

BLACK 2 votes "for" ; 0 votes "against"

In other words, the final tally is

BLUE 1

GREEN 0

RED 0

BLACK 2

8. Write-Ins

Most candidate elections allow voters to "write-in" a candidate instead of choosing a pre-registered candidate. This, often required, feature presents a unique challenge to election schemes. Here are solutions to the write-in problem.

8.1 Preregistered Write-Ins

This scheme requires that all potential write-in candidates be registered with the election authority in advance of the election. A valid registration request consists of the candidate name, ballot (precinct) identifier and race identifier. Upon receipt of a registration request, the election authority generates the next candidate number for the indicated race and computes a unique encryption generator. The candidate name, ballot identifier, race identifier, candidate number and generator are stored in a secure data area. Prior to the start of the election, the registration service is closed and the information for all write-in candidates is merged into the appropriate precinct ballots.

The primary advantage of the pre-registration model is that all write-in candidate names appear on all ballots, and everyone casts a vote for or against every write-in candidate. Once the election begins, there is no difference between a write-in candidate and a candidate who is officially qualified to be on the ballot. Thus, write-in

candidate votes receive all of the security advantages inherent in the homomorphic encryption scheme. No information can leak about who voted for a particular write-in candidate and no special steps are required during the ballot submission half of the protocol.

There are two major disadvantages to the pre-registration model. First, the pre-registration model is contrary to the way write-ins are currently handled in those jurisdictions where they are allowed. To use our scheme, the jurisdiction must decide to allow registration of write-in candidates prior to the election and to disallow submission of new write-in candidate names during the election. The resultant paradigm change may present political or legislative problems in some areas. Second, the registration process must sever the link between a write-in candidate name and the person submitting the name. Although such a linkage is not a guarantee that the person submitting the name will actually vote for the candidate, the probability is high - especially if the candidate receives only one vote. In any case, it is likely that voters will be uncomfortable with election officials knowing who registered write-in candidate names. A secure anonymous channel can be used to sever the link. However, this may expose the write-in registration system to denial-of-service-attacks, where the server is bombarded with registration requests. This denial-of-service attack will slow service and/or make the number of write-in candidates unmanageable or infeasible.

The preferred implementation would use a verifiable anonymous protocol, such as a blind signature protocol. Permission to register a write-in candidate can then be restricted to registered voters, while still severing the link between the write-in candidate and the person registering the write-in. This does not entirely eliminate the possibility of malicious attacks, but does limit the damage to repeated "pinging" of the registration server (which is a vulnerability of any Internet-based service). If each registered voter is restricted to obtaining one authorization number through the blinding protocol, and each authorization number is restricted to submitting one write-in candidate name per race.

8.2 Write-In Database

This scheme requires a pre-existing database on an election server containing a record for each person eligible to hold any office appearing on the ballot.

The record contains the person's name, unique identifier (*e.g.*, social security number) and an encryption generator. For any given race, the voter may fill in the name of a write-in candidate. The database is queried for that name, and if a match is found, the unique identifier and any necessary encryption data are used to form the vote for that candidate.

Since the write-in candidate identifier is plainly visible on the encrypted ballot, any observer can conclude with high confidence that the person submitting the encrypted ballot voted *yes* for that write-in candidate. The conclusion is not 100% certain because a voter can submit a *no* vote for any write-in candidate appearing in the database. Under normal circumstances, this would be very unlikely. However, we can increase the uncertainty in several ways. First, we require submission of a write-in vote for all races. If the voter does not wish to vote for a write-in candidate for a particular race, then a random candidate name is selected from the database and a *no* vote is cast for that candidate. Second, we further increase the uncertainty by placing the names of all write-in candidates selected so far into a pool from which subsequent random candidate names will be selected. This dramatically increases the chances that all write-in names, including the "real" ones, will be repeated in the ballots of multiple voters. Thus, information about who voted for a write-in candidate is concealed. The primary advantage to this scheme is that casting a vote for a write-in candidate is virtually identical to the traditional method. Write-in candidates need not be registered prior to the election - they can be selected during the election. Another potential advantage is that the database of all valid write-in candidates will help prevent misspellings and the casting of votes for bogus candidates.

There are three major disadvantages of this method. First, the creation, maintenance and access of the database may not be practically feasible. For a national election, the database would have to contain the names of all U.S. citizens eligible for holding any office in the country. This would probably result in well over 100 million records of at least 50 bytes each, or 5 gigabytes of data. For local elections, it may be sufficient to have a database containing all citizens residing in the state who are above a certain age. This could still result in 500 megabytes of data or more for large states. There are significant issues surrounding the problems of obtaining the data and keeping it

current. If a name is not in the database, it cannot be used for a write-in vote. If the name has been omitted in error, there could be severe political fallout.

Second, the database queries require a secure anonymous channel. If the channel is not anonymous, the election authorities will be able to correlate an inquiry by name with a vote for the matching write-in candidate. In fact, an inquiry by name is itself a pretty good indicator that the voter will cast a yes vote for that write-in candidate. It would also be possible to do a correlation by eliminating all voters who made a random query to obtain a fake write-in name. Some concealment is possible by forcing all voters to make both a random inquiry and an inquiry by name. After at least one person votes, the names in the random selection list can be used for inquiries by name. However, it will be known to the authority whether the first voter did a random inquiry or an inquiry by name, and we cannot conceal the voter who made the first inquiry for a particular name. Once again, if a secure anonymous channel is used to prevent these leaks, then we open the database server to denial-of-service attacks. Thus, a verifiable anonymous protocol such as blind signatures is preferred.

Third, although forcing everyone to vote for or against a write-in candidate in every race provides some concealment, there are several unavoidable information leaks at the boundary conditions. There is a very high probability, almost a certainty, that the first ballot submitted for a write-in candidate that receives at least one *yes* vote contains a *yes* vote for that candidate. One possible solution is to conceal the arrival order of the ballots from everyone except the election authority by using a secure channel for casting votes, delaying publication of the votes until the polls close, and then shuffling the votes prior to publication on the bulletin board (or, they can simply be posted in alphabetical or numeric order).

Unfortunately, the fake write-in and shuffling schemes cannot conceal the fact that the last voter voted *yes* for a new write-in candidate. Finally, if the number of ballots containing the same write-in candidate number is close to the number of *yes* votes for that candidate, the pool of persons voting for that candidate can be identified with enough certainty to be potentially dangerous to that group. This is likely to happen in small jurisdictions and may happen as an accidental consequence of our random selection process. One way to plug the information leaks in this scheme is to encrypt the write-in candidate numbers. That way, no one can identify who voted for or against any given

write-in candidate. However, the candidate number must be decrypted in order to perform the proof of validity and to determine the bin to which the encrypted vote should be added. One protocol would use standard public key encryption techniques: the candidate number would be encrypted with a randomly generated session key, which would in turn be encrypted with a shared public key. Another copy of the session key would be encrypted with the voter's public key for later use in verifying the candidate number in the optional voter verification process.

When the polls close, write-in votes are forwarded to a special set of authorities called *write-in authorities*. The write-in authorities use a shared-key threshold scheme to decrypt the candidate numbers, such that at least t authorities would have to conspire to reveal the connection between a voter and his/her write-in candidate numbers. The write-in authorities perform the proof of validity for each write-in ballot and add it to the encrypted tally for that candidate. It is important to note that the write-in authorities are distinct from the *tallying authorities* and *do not* possess shares of the key used to decrypt the final tally. However, there is no protection against conspiracy among tallying authorities and write-in authorities.

Although this approach has privacy advantages to the voter that casts a write-in candidate, it suffers from reduced accountability. Once the candidate number is masked, the audit trail is broken. An independent observer cannot verify that the vote was properly credited to the correct candidate because he/she cannot see the candidate number through the encryption on the original ballot. Therefore, the observer can not add up all the encrypted votes for a particular write-in candidate and verify that the it matches the encrypted tally for that candidate. Only the second set of authorities can do that, which is a weak form of verifiability.

8.3 Write-in Proof of Validity

In this section we show how it is possible to modify the voter proof of validity discussed in section 1 so that it may incorporate write-in votes. The methods of the previous subsections suffer from the drawback that no method for voter proof of validity is provided. This means that in order for the vote collection authority to accept a write-in response to a question, it must either accept on faith that the rest of the voter's

ballot is constructed properly (*i.e.* is well formed), or it must decrypt the individual decisions and verify ballot validity by inspection of the (unencrypted) contents. The former opens the election to significant voter fraud (a voter might, for example, cast 1,000,000 votes for one "non-write-in" candidate, or cast a votes for more than d candidates). The latter eliminates the voter privacy associated with the basic homomorphic protocol. (See section 1 for definition of d.)

On the other hand, the proof of validity variation outlined here allows the election to conducted with the full auditability and security properties of the multi-way election protocol described in section 4, when considering only the tally of (yes/no) decisions. The tally of write-ins must be handled by decrypting the individual write-ins, and for this reason, *the write-in tally* can not have the desirable properties of universal verifiability *and* voter privacy *simultaneously*. The organization can choose whether to emphasize privacy or external auditability. In either case, it may often be possible to dismiss the tally of the write-in votes entirely by simply counting the total number of votes cast and comparing them to the homomorphically computed decisions tallies. It may be easy to argue that the write-in votes could not possibly affect the outcome of the election, and hence that they may be remain encrypted and untallied. In such a situation, the election is universally verified, and all votes remain private.

The key to making this work is that write-in votes are encrypted in the decisions formerly used as "don't care decisions." This is possible since ElGamal encryption allows for encryption of arbitrary messages m, not just messages of the form $G^y{}_k$ or $G^n{}_k$. A "write-in" decision is then encrypted in the form

$$(x, y) = (g^\alpha, h^\alpha v) \tag{29}$$

where v is a standard text encoding of the voter's write-in vote. The value $v = G^{nk}$ ($n = -1$ in most of our examples) is agreed upon to be a special write-in value. It is special because it means "not a vote." (This is fine in practice, since the standard encoding of G^{nk} will always be illegible in any spoken language.)

Suppose that we now have a d-of-n issue where we wish to allow write-ins. The voter, V performs the following operations:

1. V's voted ballot contains $n+d$ ElGamal encrypted values (as in section 1).
2. The first n of these must be of the form $(x,y) = (g^{\alpha} , h^{\alpha} G^{v})$ (where $v = y_k$ (1) or $v = n_k$ (-1); see sections 4.1 and 4.2.)). Again, as in section 1, these encrypt the choices "yes/no" or "for/against."
3. Assuming V has cast j "for" decisions in the previous step, V must now complete the remaining d decisions. This is done by the following rules
   a. At least $w = \max \{ 0 , d - j \}$ of these must be of the form $(x,y) = (g^{\alpha} , h^{\alpha} G^{nk})$.
   b. The remaining $d - w$ may be encrypted using equation 1 for V's chosen write-in string v.
4. V must also submit a proof of validity with his voted ballot. This consists of
   a. A proof of knowledge that "at least n of the decisions are of the form $(x,y) = (g^{\alpha} , h^{\alpha} G^{nk})$." Proof is identical to the second half of the proof described in section 1 (except that the value n referred to there is actually $n+d$ in this case).
   b. For each of the first n encrypted decisions separately, V constructs a proof of knowledge that it is of the form $(x,y) = (g^{\alpha} , h^{\alpha} G^{v})$, where $v = y_k$ (1) or $v = n_k$ (-1). Each of these proofs can be seen as a 1-of-1 proof as per section 1. (Alternatively, by applying the work of Cramer, Damgrd and Shoenmakers, directly, one can shorten proofs of this type by a factor of 2.)

The vote collection center performs the following operations:

1. Verify each of the voter's separate proofs described above. The techniques for doing this are described in section 1. (Notice that these proofs taken together ensure that
   a. The n "homomorphic" decisions were each valid - that is they were not encryptions of multiple votes.

b. The sum of the number of "for" homomorphic decisions cast by the voter, and the number of write-in decisions that were not "not-a-vote" is at most d. (Remember that an encryption of G^{nk} in a write-in decision is interpreted as "not-a-vote.")

2. Aggregates and decrypts encrypted "homomorphic" decisions (first n in the voter's voted ballot) as in section 4.

3. Individually decrypts and tallies the remaining write-in decisions (last d in the voter's voted ballot). This last step may be shared so as to functionally protect voter privacy. Full publication of these results will make the election universally verifiable, but sacrifice voter privacy.

One skilled in the art will appreciate that the concepts of the invention can be used in various environments other than the Internet. For example, the concepts can be used in an electronic mail environment in which electronic mail ballots or forms are processed and stored. In general, a web page or display description (*e.g.*, the bulletin board) may be in HTML format, email format, or any other format suitable for displaying information (including character/code based formats, bitmapped formats and vector based formats). Also, various communication channels, such as local area networks, wide area networks, or point-to-point dial-up connections, may be used instead of the Internet. The various transactions may also be conducted within a single computer environment, rather than in a client/server environment. Each voter or client computer may comprise any combination of hardware or software that interacts with the server computer or system. These client systems may include television-based systems, Internet appliances and various other consumer products through which transactions can be performed.

In general, as used herein, a "link" refers to any resource locator identifying a resource on the network, such as a display description of a voting authority having a site or node on the network. In general, while hardware platforms, such as voter computers, terminals and servers, are described herein, aspects of the invention are equally applicable to nodes on the network having corresponding resource locators to identify such nodes.

Unless the context clearly requires otherwise, throughout the description and the claims, the words 'comprise', 'comprising', and the like are to be construed in an

inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to". Words using the singular or plural number also include the plural or singular number, respectively. Additionally, the words "herein", "hereunder", and words of similar import, when used in this application, shall refer to this application as a whole and not to any particular portions of this application.

The above description of illustrated embodiments of the invention is not intended to be exhaustive or to limit the invention to the precise form disclosed. While specific embodiments of, and examples for, the invention are described herein for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. The teachings of the invention provided herein can be applied to other encryption applications, not necessarily the electronic voting system described above.

The various embodiments described above can be combined to provide further embodiments. All of the above references and U.S. patents and applications are incorporated herein by reference. Aspects of the invention can be modified, if necessary, to employ the systems, functions and concepts of the various patents and applications described above to provide yet further embodiments of the invention.

These and other changes can be made to the invention in light of the above detailed description. In general, in the following claims, the terms used should not be construed to limit the invention to the specific embodiments disclosed in the specification and the claims, but should be construed to include all encryption systems and methods that operate under the claims to provide data security. Accordingly, the invention is not limited by the disclosure, but instead the scope of the invention is to be determined entirely by the claims.

CLAIMS

1. An electronic voting system for use with a computerized network, comprising:

a plurality of voting computers coupled to the computerized network, wherein each voting computer provides an electronic encrypted ballot representing N number of votes, wherein each vote is a choice between two options and wherein N is greater than one, wherein each electronic ballot is encrypted under a discrete log asymmetric encryption process using underlying groups Z_p or elliptic curve, and wherein each electronic ballot includes a non-interactive proof of validity that ensures the electronic ballot is well-formed;

at least first, second and third authority computers coupled to the computerized network that each include first, second and third cryptographic keys, respectively, and wherein at least two of the first, second and third keys are necessary for decrypting at least a portion of the plurality of electronic ballots, wherein the electronic ballots were encrypted under a threshold asymmetric encryption process; and

a server computer system coupled to the computerized network, wherein the server computer system is configured to: receive the plurality of electronic ballots from the plurality of voting computers; verify the proof of validity of each of the plurality of electronic ballots; form an encrypted tally of the votes from the plurality of electronic ballots; transmit the encrypted tally to the first, second and third authority computers; receive ballot decryption shares produced with the at least two of the first, second and third cryptographic keys and the encrypted tally by the first, second and third authority computers; and compute a decrypted tally based on the encrypted tally and the decryption shares.

2. The system of claim 1, further comprising:

at least one voting poll computer coupled to the computerized network and providing another plurality of electronic encrypted ballots to the server computer system.

3. The system of claim 1, further comprising:

at least one voting poll computer coupled to the computerized network, wherein the voting poll computer is coupled to a plurality of additional terminals over a network to receive, and provide to the server computer system, another plurality of electronic encrypted ballots.

4. The system of claim 1 wherein the computerized network includes the World Wide Web, wherein each of the plurality of voting computers include a web browser program, and wherein the server computer system includes:

at least two web server computers, each having at least one of the data storage devices, wherein at least one of the web server computers provides a ballot box web page for the plurality of voting computers to post their respective electronic ballots thereto.

5. The system of claim 1 wherein the plurality of voter computers include at least one palm-sized computer, cell phone, wearable computer, interactive television terminal or Internet appliance.

6. A computer system for receiving a plurality of electronic ballots, comprising:

a server computer configured to:

receive a plurality of electronic ballots representing a multi-way election, wherein the multi-way election represents a choice of a d number of options selected from an N number of options, wherein N and d are greater than one and N is greater than d, and wherein each electronic ballot represents the multi-way election as N number of votes with each vote being a choice between two unequal choices, and wherein each electronic ballot is encrypted under a discrete log asymmetric encryption process using an underlying mathematical group;

form an encrypted tally of the votes from the plurality of electronic ballots without decrypting any of the N number of votes on each of the plurality of electronic ballots; and

compute a decrypted tally based on the encrypted tally without decrypting any of the N number of votes on each of the plurality of electronic ballots.

7. The system of claim 6 wherein the electronic ballots are encrypted using Z_p or elliptic curve groups and include a non-interactive proof of validity that ensures the electronic ballot is well-formed, wherein the server computer receives the electronic ballots from a communication network, and wherein the server computer is configured to:

verify the proof of validity of each of the plurality of electronic ballots;

transmit over the network the encrypted tally to authority computers;

receive ballot decryption shares produced by at least a subset of authority cryptographic keys and the encrypted tally; and

decrypt the encrypted tally using the received ballot decryption shares.

8. The system of claim 6 wherein the server computer is configured to:

compute a series of base values for a log under the discrete log encryption process up to a predetermined amount corresponding to an upper bound on a total number of votes from the plurality of electronic ballots;

store in memory the computed series of base values in a table before computing the decrypted tally; and

decrypt the encrypted tally by comparing the stored values to the encrypted tally to produce the decrypted tally.

9. The system of claim 6 wherein the electronic ballots include a non-interactive proof of validity, and wherein the server computer is configured to verify the proof of validity of each of the plurality of electronic ballots.

10. The system of claim 6 wherein the electronic ballots include a non-interactive proof of validity that ensures the electronic ballot includes only N number of votes and d number of options without revealing which d of N options are selected, and wherein the server computer is configured to verify the proof of validity of each of the plurality of electronic ballots.

11. The system of claim 6 wherein the electronic ballots include write-in candidates.

12. The system of claim 6 wherein the server computer is coupled to the Internet and receives therefrom the electronic ballots.

13. The system of claim 6 wherein the wherein the server computer is configured to:

transmit over the network the encrypted tally to authority computers;

receive ballot decryption shares produced by at least a subset of authority cryptographic keys and the encrypted tally; and

decrypt the encrypted tally using the received ballot decryption shares.

14. The system of claim 6 wherein the electronic ballots are encrypted using a ring of integers, having a modulus integer value p.

15. The system of claim 6 wherein the electronic ballots are encrypted using an elliptic curve group.

16. The system of claim 6 wherein the server computer is configured to transmit a digitally signed receipt in response to each received electronic ballot.

17. The system of claim 6 wherein the wherein the server computer is configured to:

transmit to at least one third party vote verification authority a digitally signed receipt in response to each electronic ballot received from a voter, and wherein for each received receipt, the authority stores the receipt, adds another digital signature to the receipt and forwards the receipt to the voter.

18. An electronic voting method, comprising:

selecting a subgroup generator g selected from a group G;

generating n number of secret authority keys and a public key h based on the group G and subgroup generator g;

forming and distributing a plurality of initial electronic ballots to a plurality of voters based on the public key h, wherein the formed ballot includes N numbers of votes, wherein each vote is a choice between two options and wherein N is greater than one;

for each ballot,

encrypting each ballot under an asymmetric encryption process employing the group G, subgroup generator g, public key h, and a secret voter key, and

performing and providing results of a non-interactive proof of validity that ensures the electronic ballot is well-formed;

receiving the plurality of electronic ballots from the plurality of voters;

verifying the proof of validity of each of the plurality of electronic ballots;

forming an encrypted tally of the votes from the plurality of electronic ballots;

distributing the encrypted tally for creation of decryption shares based on the secret authority keys;

receiving the decryption shares from at least k number of the secret authority keys; and

computing a decrypted tally based on the encrypted tally and the decryption shares.

19. The method of claim 18 wherein encrypting each ballot includes encrypting each ballot using Z_p or elliptic curve groups;

and wherein distributing a plurality of initial electronic ballots, receiving the plurality of electronic ballots, distributing the encrypted tally, and receiving the decryption shares are all performed over a computer network.

20. The method of claim 18, further comprising:

determining an upper bound on a total number L of votes for the plurality of electronic ballots; and

computing and storing L^{α} number of values, wherein $0 \leq \alpha \leq 1$, and

wherein computing a decrypted tally includes comparing the stored values to the encrypted tally to produce the decrypted tally.

21. The method of claim 18 wherein performing and providing results includes providing a non-interactive proof of validity that ensures the electronic ballot includes only N number of votes and d number of options without revealing which d of N options are selected.

22. The method of claim 18 wherein the group G is Z_p.

23. The method of claim 18 wherein the group G is an elliptic curve group.

24. The method of claim 18, further comprising transmitting digitally signed receipts in response to each received electronic ballot.

25. The method of claim 18, further comprising:

transmitting digitally signed receipts in response to each received electronic ballot to at least one third party vote verification authority;

adding an authority digital signature to the receipt; and

forwarding the receipt to the voter.

26. A computer-readable medium providing instructions, when implemented by a voting computer perform an electronic voting method, comprising:

receiving an initial electronic ballot representing a multi-way election, wherein the multi-way election represents a choice of a *d* number of options selected from an N number of options, wherein N and *d* are greater than one and N is greater than *d*, and wherein the initial electronic ballot represents the multi-way election as N number of votes with each vote being a choice between two unequal choices;

receiving N number of input choices from a voter to form a voted electronic ballot;

encrypting the voted electronic ballot under a discrete log asymmetric encryption process using an underlying mathematical group; and

providing the encrypted electronic ballot to an electronic ballot collector for aggregation into an encrypted tally of received encrypted electronic ballots.

27. The computer-readable medium of claim 26 wherein encrypting the voted electronic ballot includes encrypting using Z_p or elliptic curve groups, and wherein the method includes:

performing a non-interactive proof of validity on the voted electronic ballot that ensures the electronic ballot is well-formed, and

wherein receiving the initial electronic ballot includes receiving the initial electronic ballot from a communication network, and providing the encrypted electronic ballot to an electronic ballot collector includes transmitting the encrypted electronic ballot over the network to an electronic ballot collection server computer.

28. The computer-readable medium of claim 26 wherein the method includes:

performing a non-interactive proof of validity on the voted electronic ballot that ensures the voted electronic ballot includes only N number of votes and d number of options without revealing which d of N options are selected.

29. The computer-readable medium of claim 26 wherein the underlying mathematical group is a ring of integers, having a modulus integer value p (Z_p).

30. The computer-readable medium of claim 26 wherein the underlying mathematical group is an elliptic curve group.

31. The computer-readable medium of claim 26, further comprising receiving a digitally signed receipt after providing the encrypted electronic ballot.

32. The computer-readable medium of claim 26 wherein receiving an initial electronic ballot includes receiving a digital signature with the initial electronic ballot, wherein the digital signature is from the electronic ballot collector.

33. The computer-readable medium of claim 26 wherein the computer-readable medium is an Internet connection link to the voting computer.

34. In an electronic voting method, a transmitted signal for use by an authority computer, comprising:

an encrypted tally of electronic ballots representing a multi-way election, wherein the multi-way election represents a choice of a *d* number of options selected from an N number of options, wherein N and *d* are greater than one and N is greater than *d*, wherein each electronic ballot represents the multi-way election as N number of votes with each vote being a choice between two unequal choices, and wherein the encrypted tally comprises a plurality of the electronic ballots encrypted under a discrete log asymmetric encryption process using an underlying mathematical group and a public key h, and

wherein the encrypted tally allows the authority computer to generate a decryption share based on one of a number n of secret authority keys, and wherein at least a k number decryption shares, generated by k number the secret authority keys, are necessary to compute a decrypted tally based on the encrypted tally and the k number of decryption shares.

35. The transmitted signal of claim 34 wherein each of the plurality of electronic ballots includes, before aggregation into the encrypted tally, a non-interactive proof of validity on the electronic ballot that ensures the electronic ballot includes only N number of votes and d number of options without revealing which d of N options are selected.

36. The transmitted signal of claim 34 wherein each of the plurality of electronic ballots includes, before aggregation into the encrypted tally, a non-interactive proof of validity on the electronic ballot that ensures the electronic ballot includes only N number of votes and at most d number of options without revealing which of at most d of N options are selected.

37. The transmitted signal of claim 34 wherein each of the plurality of electronic ballots includes, before aggregation into the encrypted tally, a non-interactive proof of validity on the electronic ballot that ensures the electronic ballot includes only N number of votes and d number of options without revealing which d of N options are

selected, and wherein at least one of the plurality of ballots in the encrypted tally includes a write-in candidate vote.

38. The transmitted signal of claim 34 wherein the underlying mathematical group is a ring of integers, having a modulus integer value p (Z_p), or an elliptic curve group.

39. An electronic voting method, comprising:

selecting a subgroup generator g selected from a group G and generating n number of secret authority keys and a public key h based on the group G and subgroup generator g;

receiving a plurality of electronic ballots from a plurality of voters based on the public key h, wherein the formed ballot includes N numbers of votes, wherein each vote is a choice between two options and wherein N is greater than one, and wherein each ballot is encrypted under an asymmetric encryption process employing the group G, subgroup generator g, public key h, and a secret voter key, and

forming an encrypted tally of the votes from the plurality of electronic ballots;

distributing the encrypted tally for generation of decryption shares based on the secret authority keys;

receiving decryption shares based on at least k number of the secret authority keys; and

computing a decrypted tally based on the encrypted tally and the received decryption shares.

40. The method of claim 39, further comprising:

determining an upper bound on a total number L of votes for the plurality of electronic ballots; and

computing and storing L^{α} number of values, wherein $0 \leq \alpha \leq 1$, and

wherein computing a decrypted tally includes comparing the stored values to the encrypted tally to produce the decrypted tally.

41. The method of claim 39, further comprising verifying a proof of validity corresponding to each of the received plurality of electronic ballots, wherein each proof of

validity is a non-interactive proof of validity that ensures the electronic ballot includes only N number of votes and d number of options without revealing which d of N options are selected.

42. The method of claim 39, further comprising verifying a proof of validity corresponding to each of the received plurality of electronic ballots, wherein each proof of validity is a non-interactive proof of validity that ensures the electronic ballot includes only N number of votes and at least d number of options without revealing which of at least d of N options are selected.

43. The method of claim 39, further comprising:

verifying a proof of validity corresponding to each of the received plurality of electronic ballots, wherein each proof of validity is a non-interactive proof of validity that ensures the electronic ballot includes N number of votes and d number of options without revealing which d of N options are selected, and wherein at least some of the plurality of electronic ballots include at least one of the d options as a write-in candidate; and

after computing a decrypted tally, decrypting and tallying the write-in candidates.

44. The method of claim 39 wherein the group G is a ring of integers, having a modulus integer value p, or an elliptic curve group.

45. The method of claim 39, further comprising transmitting digitally signed receipts in response to each received electronic ballot.

46. The method of claim 39, further comprising:

transmitting digitally signed receipts in response to each received electronic ballot to at least one third party vote verification authority before the digitally signed receipt is provided to a voter.

47. The method of claim 39, further comprising transmitting electronic ballots for voting, wherein each transmitted electronic ballot includes a digital signature or output of a hash function applied to the electronic ballot.

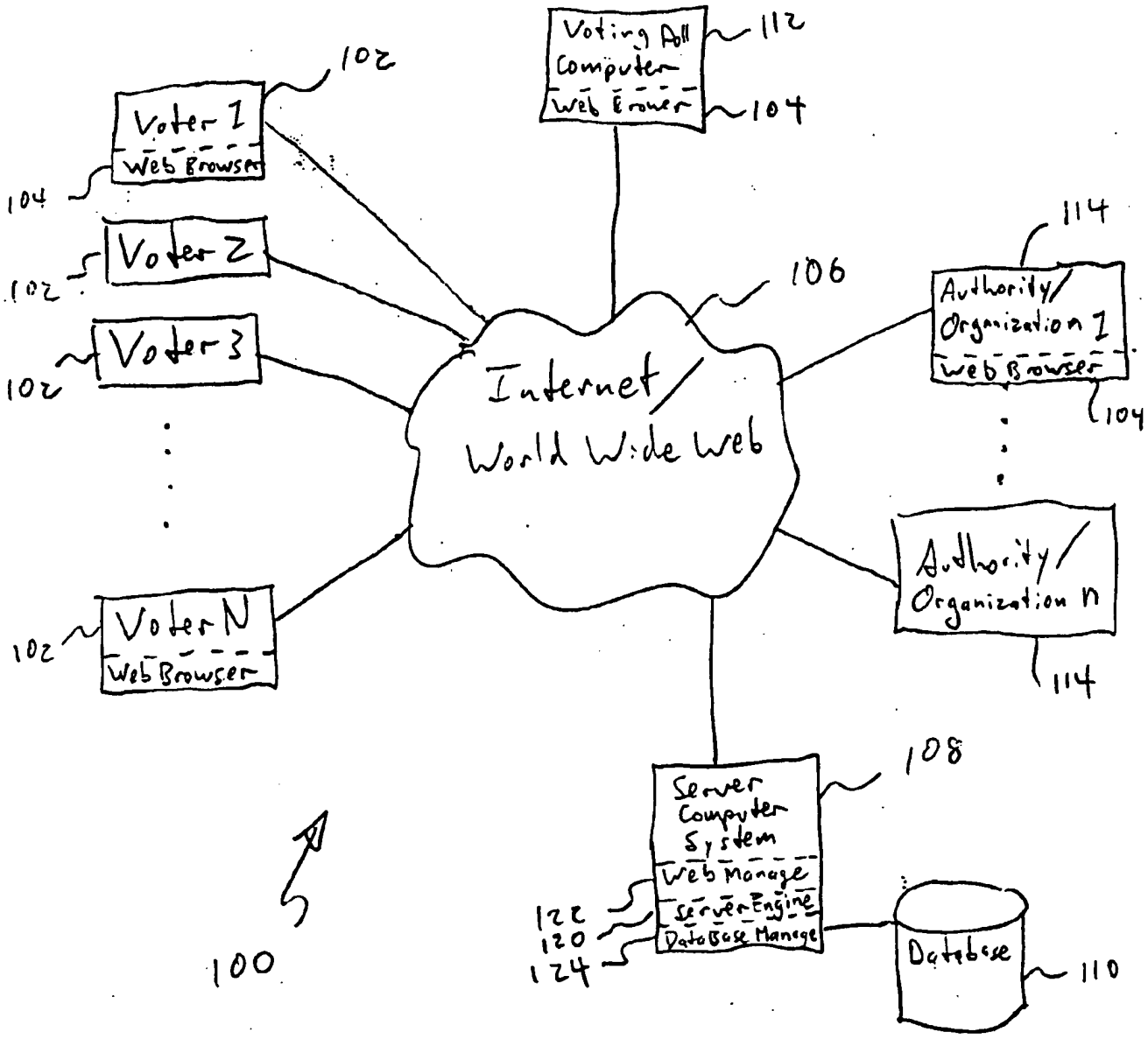

Voting Adl Computer
Web Browser
112
104
102
Voter 1
Web Browser
104
Voter 2
102
Voter 3
102
Voter N
Web Browser
102
Internet / World Wide Web
106
Authority / Organization 1
Web Browser
114
104
Authority / Organization n
114
Server Computer System
Web Manage
Server Engine
Database Manage
108
122
120
124
Database
110
100

FIG. 1

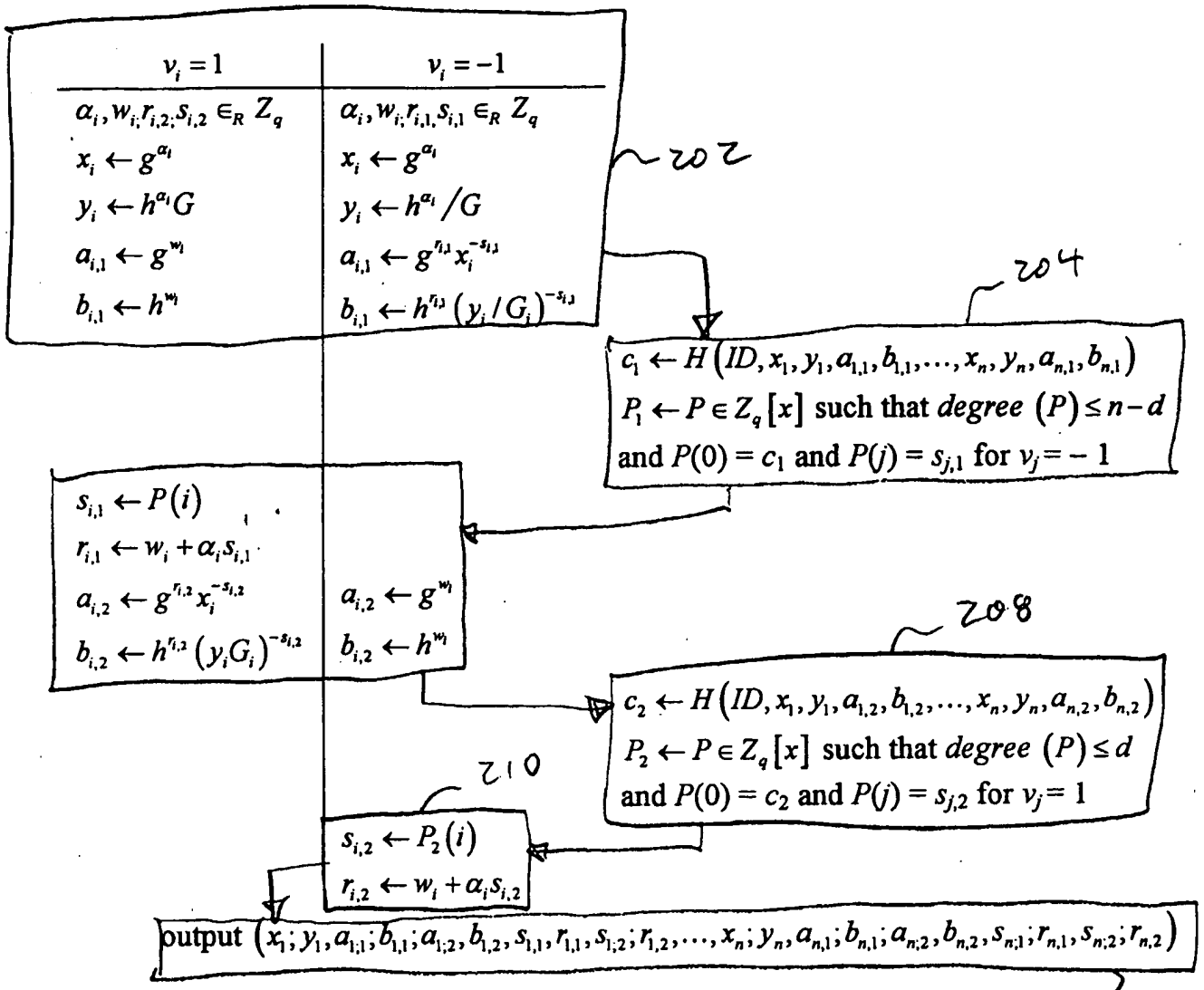


Figure 2: Ballot Encryption and Proof of Ballot Validity

200

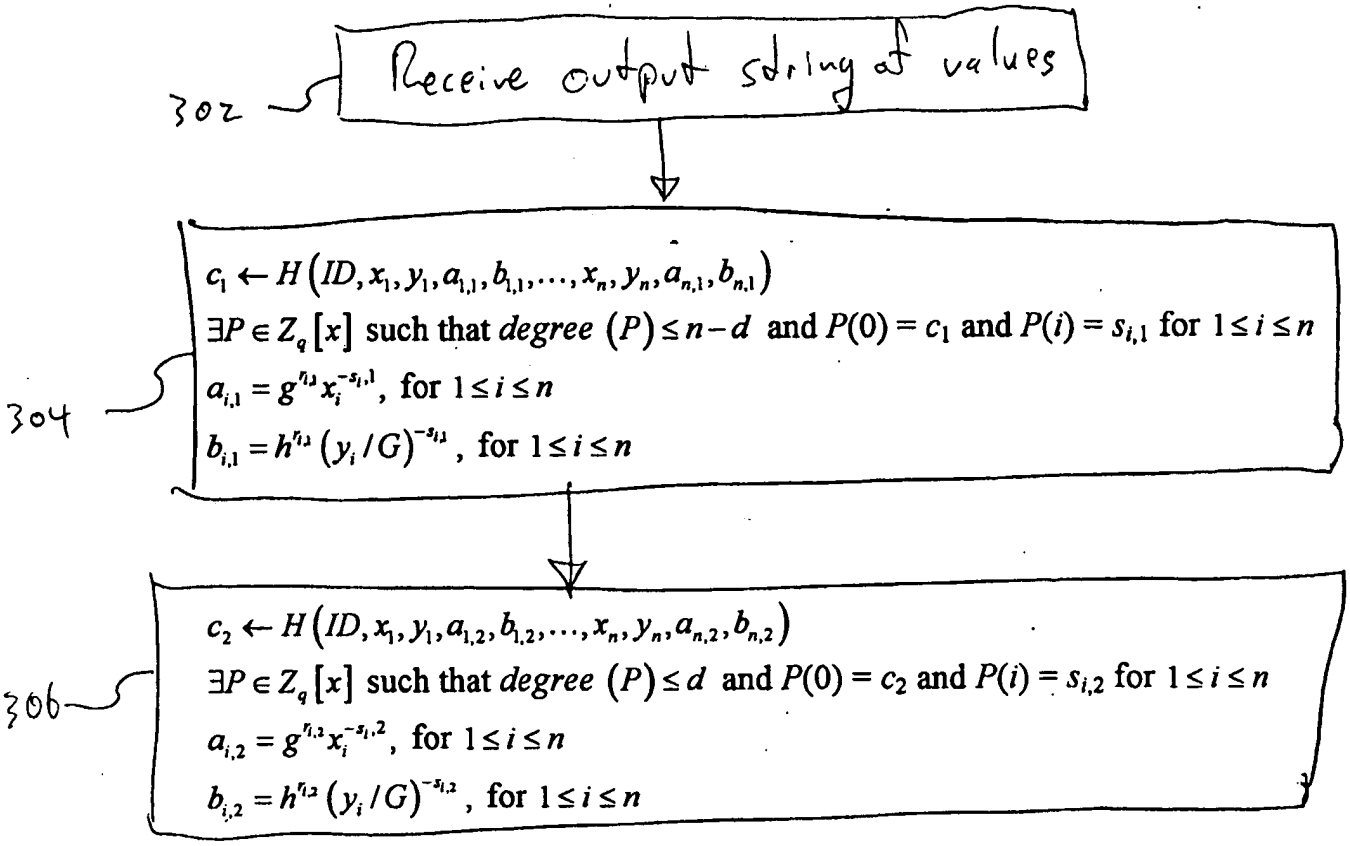


Figure 3: Verification of Ballot Validity

300

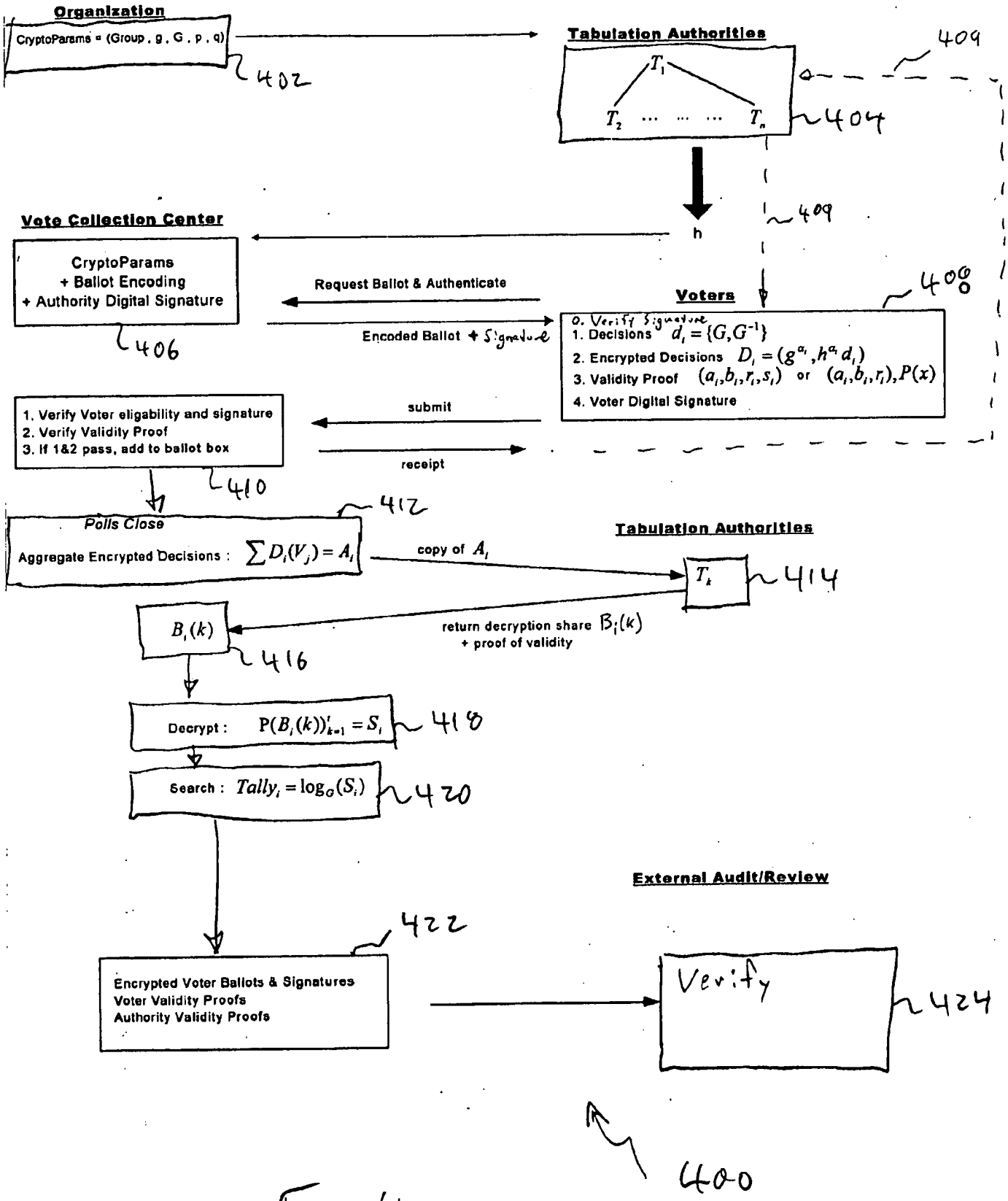

Organization
CryptoParams = (Group , g , G , p , q)
402
Tabulation Authorities
T_1
T_2 ... ... T_n
404
409
h
Vote Collection Center
CryptoParams
+ Ballot Encoding
+ Authority Digital Signature
406
Request Ballot & Authenticate
Encoded Ballot + Signature
Voters
406
0. Verify Signature
1. Decisions $d_i = \{G, G^{-1}\}$
2. Encrypted Decisions $D_i = (g^{a_i}, h^{a_i} d_i)$
3. Validity Proof (a_i, b_i, r_i, s_i) or $(a_i, b_i, r_i), P(x)$
4. Voter Digital Signature
1. Verify Voter eligability and signature
2. Verify Validity Proof
3. If 1&2 pass, add to ballot box
410
submit
receipt
Polls Close
412
Aggregate Encrypted Decisions : $\sum D_i(V_j) = A_i$
copy of A_i
Tabulation Authorities
T_k
414
$B_i(k)$
416
return decryption share $B_i(k)$
+ proof of validity
Decrypt : $P(B_i(k))_{k=1}^{l} = S_i$
418
Search : $Tally_i = \log_G(S_i)$
420
External Audit/Review
422
Encrypted Voter Ballots & Signatures
Voter Validity Proofs
Authority Validity Proofs
Verify
424
400

FIG 4

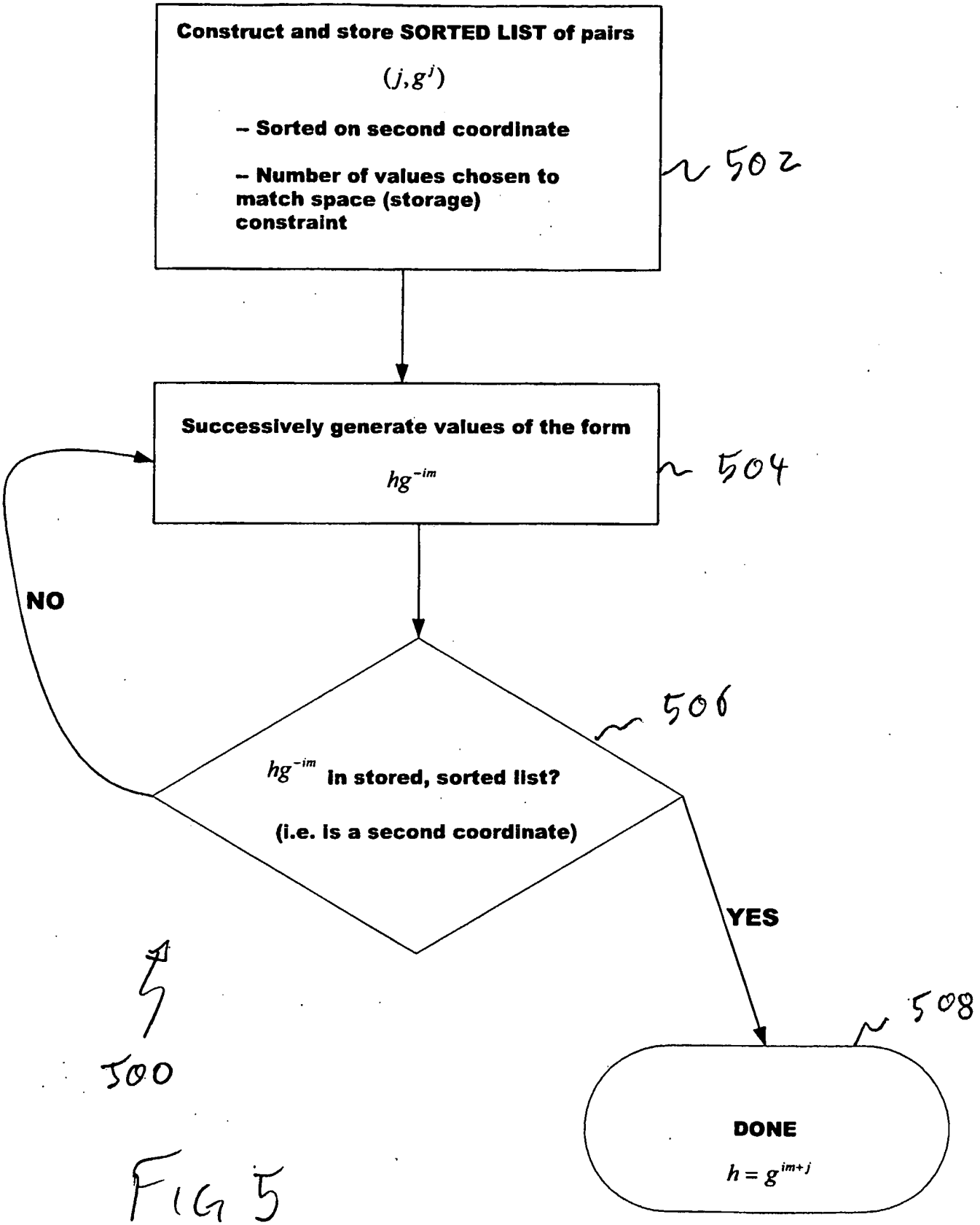

Construct and store SORTED LIST of pairs
(j, g^j)
-- Sorted on second coordinate
-- Number of values chosen to match space (storage) constraint
502
Successively generate values of the form
hg^{-im}
504
NO
hg^{-im} in stored, sorted list?
(i.e. is a second coordinate)
506
YES
DONE
$h = g^{im+j}$
508
500

FIG 5

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 March 2001 (22.03.2001)

PCT

(10) International Publication Number
WO 01/20562 A3

(51) International Patent Classification[7]: G07C 13/00, H04L 9/32

(21) International Application Number: PCT/US00/07737

(22) International Filing Date: 24 March 2000 (24.03.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/126,080 25 March 1999 (25.03.1999) US
60/149,621 16 August 1999 (16.08.1999) US

(71) Applicant *(for all designated States except US)*: **VOTE-HERE, INC.** [US/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US).

(72) Inventors; and

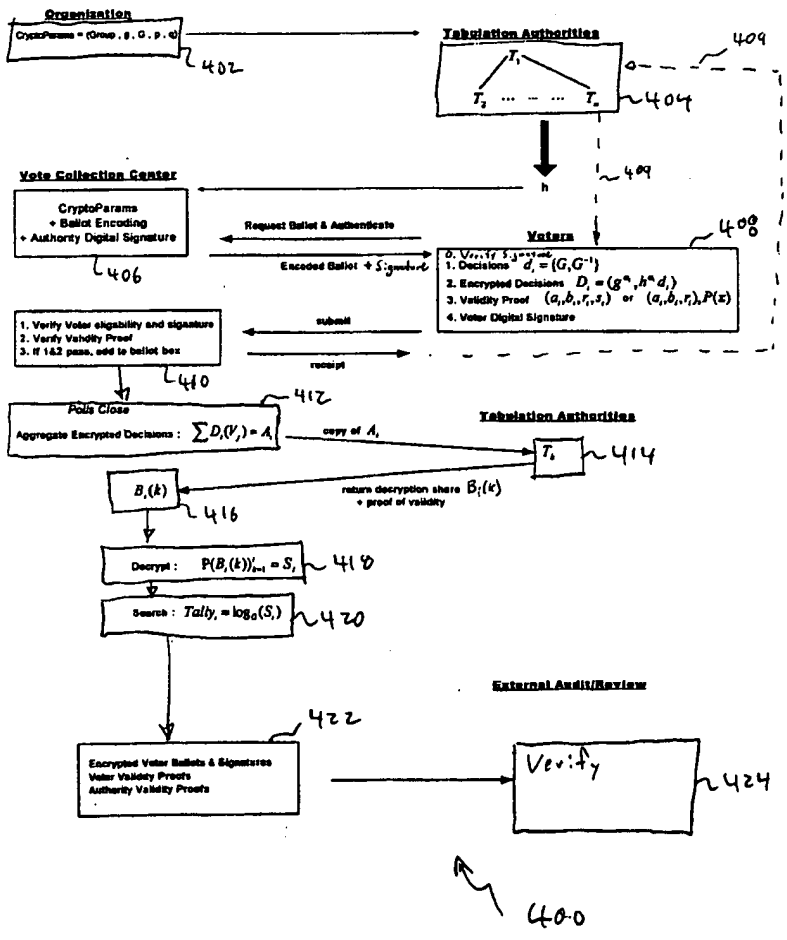
(75) Inventors/Applicants *(for US only)*: **ADLER, Jim** [—/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US). **GREEN, Richard, L.** [—/US]; 190 Lyme Road, Hanover, NH 03755 (US). **NEFF, C., Andrew** [—/US]; Suite 250, 3101 Northup Way, Bellevue, WA 98004 (US). **DAI, Wei** [—/US]; 13440 SE 24th Street, Bellevue, WA 98005 (US).

(74) Agents: **DALEY-WATSON, Christopher, J.** et al.; Perkins Coie LLP, P.O. Box 1247, Seattle, WA 98111-1247 (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE,

[Continued on next page]

(54) Title: MULTIWAY ELECTION METHOD AND APPARATUS



(57) Abstract: A scheme or protocol for computationally secure, practical electronic voting is disclosed that supports multiple ballot options and write-in options. The scheme is based on homomorphic threshold cryptographic schemes. The scheme is efficient in computing election tallies for multi-way races (*i.e.*, for choosing *d*-of-*N* options). In addition to implementation in the discrete log context, the scheme is presented in an elliptic curve setting.

WO 01/20562 A3

SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
18 October 2001

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Internat. Application No
PCT/US 00/07737

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G07C13/00 H04L9/32

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G07C H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ, IBM-TDB, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CRAMER R ET AL: "A SECURE AND OPTIMALLY EFFICIEN MULTI-AUTHORITY ELECTION SCHEME" EUROPEAN TRANSACTIONS ON TELECOMMUNICATIONS,IT,EUREL PUBLICATION, MILANO, vol. 8, no. 5, 1 September 1997 (1997-09-01), pages 481-490, XP000720074 ISSN: 1124-318X cited in the application abstract page 482, paragraph 1.2 -page 487, left-hand column, last line --- -/--	1-10, 12-14, 17-19, 21,22, 24-29, 31, 33-39, 42,44-46

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 April 2001	04/05/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Buron, E

2

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Internat. Application No
PCT/US 00/07737

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 495 532 A (SAKO KAZUE ET AL) 27 February 1996 (1996-02-27) abstract; figures 3-5,10 column 5, line 60 -column 7, line 37 column 11, line 53 - line 61	1,6,18, 26,34,39
A	BENALOH & YUNG: "Distributing the Power of a Government to Enhance the Privacy of Voters" PROC. ANNUAL ACM SYMP. ON PRINCIPLES OF DISTRIBUTED COMPUTING,XX,XX, August 1986 (1986-08), pages 52-62, XP002099136 cited in the application page 55, left-hand column, line 3 -page 56, left-hand column, line 20 page 57, paragraph 3.6	18,34,39
A	DATABASE INSPEC 'Online! INSTITUTE OF ELECTRICAL ENGINEERS, STEVENAGE, GB; CRAMER R ET AL: "Multi-authority secret-ballot elections with linear work" Database accession no. 5323491 XP000992684 cited in the application abstract & ADVANCES IN CRYPTOLOGY - EUROCRYPT '96. INTERNATIONAL CONFERENCE ON THE THEORY AND APPLICATION OF CRYPTOGRAPHIC TECHNIQUES. PROCEEDINGS, ADVANCES IN CRYPTOLOGY - EUROCRYPT '96, SARAGOSSA, SPAIN, 12-16 MAY 1996, pages 72-83, 1996, Berlin, Germany, Springer-Verlag, Germany ISBN: 3-540-61186-X	
A	DATABASE INSPEC 'Online! INSTITUTE OF ELECTRICAL ENGINEERS, STEVENAGE, GB; COHEN BENALOH J: "Secret sharing homomorphisms: keeping shares of a secret secret" Database accession no. 3158221 XP000992685 cited in the application abstract & ADVANCES IN CRYPTOLOGY - CRYPTO '86 PROCEEDINGS, SANTA BARBARA, CA, USA, 11-15 AUG. 1986, pages 251-260, 1987, Berlin, West Germany, Springer-Verlag, West Germany ISBN: 3-540-18047-8	

2

INTERNATIONAL SEARCH REPORT

Internat. l Application No
PCT/US 00/07737

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5495532 A	27-02-1996	EP 0697776 A	21-02-1996
		JP 8063533 A	08-03-1996